

Danillo Roberto Pereira

**“Fitting 3D Deformable Biological Models to  
Microscope Images”**

*“Alinhamento de Modelos Tridimensionais Usando  
Imagens de Microscopia”*

CAMPINAS  
2013





University of Campinas  
Institute of Computing



*Universidade Estadual de Campinas  
Instituto de Computação*

Danillo Roberto Pereira

**“Fitting 3D Deformable Biological Models to  
Microscope Images”**

Supervisor: Prof. Dr. Jorge Stolfi  
*Orientador(a):*

***“Alinhamento de Modelos Tridimensionais Usando  
Imagens de Microscopia”***

PhD Thesis presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a PhD degree in Computer Science.

*Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Doutor em Ciência da Computação.*

THIS VOLUME CORRESPONDS TO THE FINAL VERSION OF THE THESIS DEFENDED BY DANILLO ROBERTO PEREIRA, UNDER THE SUPERVISION OF PROF. DR. JORGE STOLFI.

*ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA POR DANILLO ROBERTO PEREIRA, SOB ORIENTAÇÃO DE PROF. DR. JORGE STOLFI.*

  
\_\_\_\_\_  
Supervisor's signature / Assinatura do Orientador(a)

CAMPINAS  
2013

Ficha catalográfica  
Universidade Estadual de Campinas  
Biblioteca do Instituto de Matemática, Estatística e Computação Científica  
Ana Regina Machado - CRB 8/5467

P414f Pereira, Danilo Roberto, 1984-  
Fitting 3D deformable biological models to microscope images / Danilo Roberto Pereira. – Campinas, SP : [s.n.], 2013.

Orientador: Jorge Stolfi.  
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Abordagem multiescala. 2. Otimização não-linear. 3. Modelos tridimensionais. I. Stolfi, Jorge, 1950-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** Alinhamento de modelos biológicos tridimensionais deformáveis usando imagens de microscopia

**Palavras-chave em inglês:**

Multiscale approach  
Non-linear optimization  
3D models

**Área de concentração:** Ciência da Computação

**Titulação:** Doutor em Ciência da Computação

**Banca examinadora:**

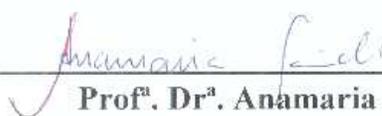
Jorge Stolfi [Orientador]  
Luiz Henrique de Figueiredo  
Marco Antônio Piteri  
Anamaria Gomide  
Hélio Pedrini

**Data de defesa:** 16-08-2013

**Programa de Pós-Graduação:** Ciência da Computação

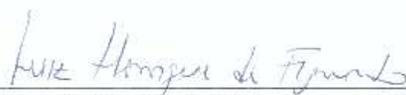
## TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 16 de Agosto de 2013, pela Banca  
examinadora composta pelos Professores Doutores:



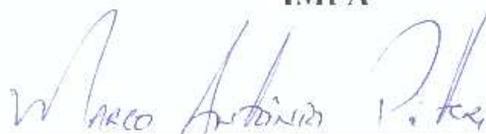
---

**Prof.ª. Dr.ª. Anamaria Gomide**  
IC / UNICAMP



---

**Prof. Dr. Luiz Henrique de Figueiredo**  
IMPA



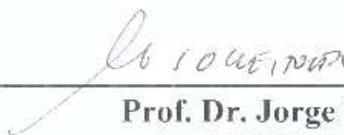
---

**Prof. Dr. Marco Antônio Piteri**  
FCT / UNESP



---

**Prof. Dr. Hélio Pedrini**  
IC / UNICAMP



---

**Prof. Dr. Jorge Stolfi**  
IC / UNICAMP



# Fitting 3D Deformable Biological Models to Microscope Images

Danillo Roberto Pereira<sup>1</sup>

August 16, 2013

**Examiner Board/*Banca Examinadora*:**

- Prof. Dr. Jorge Stolfi (*Advisor*)
- Luiz Henrique de Figueiredo  
IMPA
- Marco Antônio Piteri  
FCT - UNESP
- Anamaria Gomide  
Insituto de Computação - UNICAMP
- Hélio Pedrini  
Insituto de Computação - UNICAMP

---

<sup>1</sup>Suporte financeiro de: FAPESP 2010-2013 (Processo: 2009/13744-2), CAPES 2009-2010



# Resumo

Nesta tese descrevemos um algoritmo genérico (que denominamos MSFIT) capaz de estimar a pose e as deformações de modelos 3D de estruturas biológicas (bactérias, células e etc) em imagens obtidas por meio de microscópios óticos ou de varredura eletrônica. O algoritmo usa comparação multi-escala de imagens utilizando uma métrica sensível ao contorno; e um método original de otimização não-linear. Nos nossos testes com modelos de complexidade moderada (até 12 parâmetros) o algoritmo identifica corretamente os parâmetros do modelo em 60-70% dos casos com imagens reais e entre 80-90% dos casos com imagens sintéticas.



# Abstract

In this thesis we describe a generic algorithm (which we call MSFIT) able to estimate the pose and deformations of 3D models of biological structures (bacteria, cells, etc.) with images obtained by optical and scanning electron microscopes. The algorithm uses a image comparison metric multi-scale, that is outline-sensitive, and a novel nonlinear optimization method. In our tests with models of moderate complexity (up to 12 parameters) the algorithm correctly identifies the model parameters in 60-70 % of the cases with real images and 80-90 % of the cases with synthetic images.



# Agradecimentos

Aos meu maiores ídolos, meu pai (meu heroi) e minha mãe (minha eterna rainha, in memorian) pela educação, pelo carater e pelo carinho dado. Não tenho palavras para descrever o quanto amo vocês.

Aos professores Dr. Jorge Stolfi pela orientação, pela atenção, pelo profissionalismo, pela paciência e confiança oferecida; e principalmente pelo bom senso que teve comigo. Muitíssimo obrigado.

Aos professores do Instituto de Computação que contribuíram direta ou indiretamente na minha formação. Aos professores do curso de ciência da computação da FCT-UNESP; em especial ao prof. Dr. Marco Antônio Piteri, que me orientou durante toda graduação e que deu a oportunidade de conhecer a área acadêmica. Sou muito grato ao Sr.

Aos meus tios Márcia e Adelson, a minha avó Dair e aos meus primos Juliane e Bruno pelo apoio e por me fazer sentir mais perto dos meus pais.

A minha tia Regina e minha vó Eugênia, pela união e companherismo que tiveram comigo no momento mais difícil e delicado de minha vida.

Aos meus amigos de república pelo companherismo e por tornarem os meus dias mais agradáveis. Em especial ao Marco, ao Tiezzi, ao Rubens, ao Breves e ao Jorge.

Aos meus dois irmãos Diego e Clóvis, o primeiro de sangue e o segundo de consideração; e a minha namorada Amanda, pelo apoio incondicional.

Mesmo sabendo que a ausência só apaga sentimentos pequenos e fortalece os grandes; termino pedindo desculpa aos meus pais e a minha namorada pela minha ausência.

Eu gostaria de agradecer a FAPESP 2010-2013 (Processo: 2009/13744-2) e a CAPES (2009-2010) pelo suporte financeiro.



# Contents

<b>Resumo</b>	<b>ix</b>
<b>Abstract</b>	<b>xi</b>
<b>Agradecimentos</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation and statement of the problem . . . . .	1
1.2 Characteristics of microscopic images . . . . .	3
1.3 Computational challenges . . . . .	5
1.3.1 The exponential cost barrier . . . . .	5
1.4 Multiscale alignment strategy . . . . .	5
1.5 Organization of the thesis . . . . .	6
<b>2 Related work</b>	<b>7</b>
2.1 Specific methods . . . . .	7
2.1.1 Facial recognition . . . . .	7
2.1.2 Medical images . . . . .	8
2.1.3 Hands . . . . .	8
2.1.4 Multiple image fitting . . . . .	9
2.2 Generic methods . . . . .	9
2.2.1 Model-free methods . . . . .	9
2.2.2 Rigid methods . . . . .	9
2.3 Deformable models . . . . .	10
2.4 Tools . . . . .	10
2.4.1 Metrics . . . . .	10
2.4.2 Optimization methods . . . . .	10
2.4.3 Modeling techniques . . . . .	10



<b>I</b>	<b>Fundamental concepts</b>	<b>13</b>
<b>3</b>	<b>Fundamental concepts</b>	<b>15</b>
3.1	The organism model . . . . .	15
3.2	Formal statement of the problem . . . . .	15
3.3	Rendering a model . . . . .	16
<b>4</b>	<b>Discrepancy functions</b>	<b>17</b>
4.1	Euclidean discrepancy . . . . .	17
4.2	Normalized gradient . . . . .	18
4.2.1	Comparison of $\text{dist}_2$ and $\text{dist}_{\nabla 2}$ . . . . .	19
4.3	Multiscale discrepancy . . . . .	19
4.3.1	Comparing single and multiscale discrepancies . . . . .	20
4.4	Effectiveness of the discrepancy functions . . . . .	24
4.4.1	Moving disk . . . . .	25
4.4.2	Rotating rectangle . . . . .	26
4.4.3	Bending rectangle . . . . .	28
4.4.4	Conclusion . . . . .	28
4.5	Granularity of discrepancy functions . . . . .	29
<b>II</b>	<b>Multiscale Fitting</b>	<b>31</b>
<b>5</b>	<b>MSFit algorithm</b>	<b>33</b>
5.1	Simplified algorithm description . . . . .	33
5.1.1	Candidate level . . . . .	34
5.1.2	Active parameters . . . . .	34
5.2	Detailed algorithm . . . . .	35
5.2.1	Hereditary discrepancy . . . . .	36
<b>III</b>	<b>Models and tests</b>	<b>39</b>
<b>6</b>	<b>Modeling technique</b>	<b>41</b>
6.1	Model instantiation . . . . .	41
6.2	Rendering procedure . . . . .	43
6.2.1	Scale of image . . . . .	43
6.2.2	Anti-aliasing . . . . .	43



<b>7</b>	<b>Test models</b>	<b>45</b>
7.1	The <code>ball</code> model . . . . .	45
7.2	The <code>box</code> model . . . . .	45
7.3	The <code>cyl</code> model . . . . .	46
7.4	The <code>volv</code> model . . . . .	47
7.5	The <code>prmc</code> model . . . . .	47
7.6	The <code>cycl</code> model . . . . .	49
7.7	The <code>eleg</code> model . . . . .	52
<b>8</b>	<b>Experimental evaluation</b>	<b>55</b>
8.1	Test datasets . . . . .	55
8.2	Automatic tests evaluation . . . . .	58
8.3	Average performance . . . . .	58
8.4	Internal parameters . . . . .	59
8.5	Results . . . . .	59
8.5.1	Effect of the distance estimator $D$ . . . . .	59
8.5.2	Effect of image scale factor reduction . . . . .	64
8.5.3	Effect of the number of rays cast per pixel . . . . .	67
8.5.4	Effect of the number of iterations . . . . .	70
8.5.5	Effect of the queue size . . . . .	74
<b>9</b>	<b>Conclusion</b>	<b>79</b>
9.1	Contributions of this thesis . . . . .	79
9.2	Future work . . . . .	80
<b>A</b>	<b>Nonlinear optimization method</b>	<b>81</b>
A.1	Function optimization by the edge-divided simplex method . . . . .	81
A.1.1	Quadratic interpolation . . . . .	81
A.1.2	Quadratic optimization . . . . .	83
A.1.3	Iterative quadratic optimization . . . . .	84
A.1.4	Examples . . . . .	85
	<b>Bibliography</b>	<b>99</b>



# Chapter 1

## Introduction

### 1.1 Motivation and statement of the problem

The popularity of digital microscopes created an acute shortage of manpower to analyze and classify their these images. In clinical microscopy examinations, for example, the analysis of a single image that is produced in a matter of seconds may demand hours of visual inspection.

In this work we propose an algorithm to automate this task. We are mainly concerned with biological structures smaller than  $\approx 1$  mm, imaged through a microscope of some sort. To simplify the discussion we will refer to such structures as *organisms*, even though our method can be applied also to isolated cells of multicellular organisms, organs, organelles, etc.

We consider here a common type of analysis which is the matching of a given image of certain organisms to an abstract model of the same. Each kind of organism, at a specific stage of life, usually has recognizable morphology. For example figures 1.1 and 1.2 show the schematic morphology of erythrocytes (red blood cells) and a of *Paramecium* (an aquatic unicellular animal).

A major difficulty in this problem is the fact that the same organism can produce very different images, depending on its orientation in space. Furthermore, many organisms of interest have gelatinous and/or elastic consistency, and may suffer considerable deformation, as can be seen in figure 1.3. They may also have visible internal structures which may be mobile and independently deformed, as well as ingested particles and externally attached debris. See figure 1.4.

The organism-matching task is made harder by the translucent nature of most microscopic biological structures, and by the complexities of the microscope imaging process — which may include very short depth-of-focus, interference and diffraction artifacts, and the common occurrence of dust and other debris over and around the organism. Yet, with

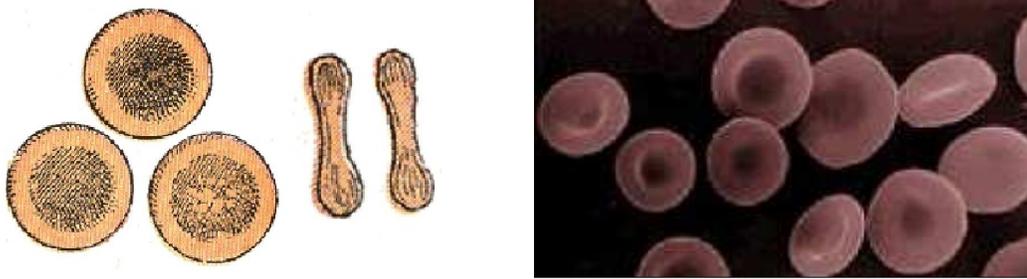


Figure 1.1: Schematic morphology of erythrocytes [1] (left) and a scanning electron microscope image of the same (right).

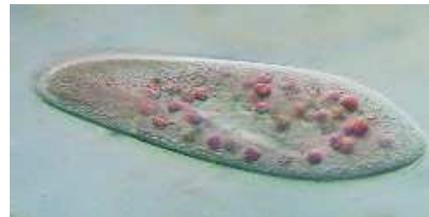
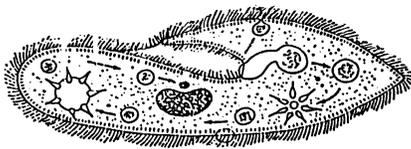


Figure 1.2: Schematic morphology of *Paramecium* sp. [1] (left) and an optical image of same [2] (right).



Figure 1.3: Optical microscope images of *Caenorhabditis elegans* ( $\approx 1\text{mm}$ ) [3, 4] showing deformation.

some experience, an human observer can identify microscopic images of the organisms of various kinds, and align them with their three-dimensional schematic diagrams like those of figure 1.1 and 1.2.

Our goal in this thesis was to replicate this human ability in software, as far as possible. Specifically, we sought to produce mathematical and innovative computational tools for



Figure 1.4: Microscopic image of a *Daphnia* [5] (left,  $\approx 1mm$ ) and *Paramecium* sp (right) [6] (right,  $\approx 0.3mm$ ), showing internal structures, ingested particles and attached debris.

the recognition and alignment of 3D biological models with 2D images obtained by optical microscopes or scanning electron microscopes.

In principle, this work has immediate applications in biology and medicine, by automating extremely laborious tasks of identification, classification and counting of organisms in microscopic images. However, the problem of matching deformable 3D models with 2D images has a much wider range of applications, including face recognition, industrial inspection of non-rigid parts, analysis of medical imaging (radiography and angiography), augmented reality and many others. Furthermore, we believe that the techniques that we developed to address specific sub-problems may be useful in other areas.

Our method may also be useful in other domains, such as face recognition, industrial inspection of non-rigid parts, analysis of radiography and angiography, augmented reality and many others.

## 1.2 Characteristics of microscopic images

There are two main microscope types commonly used in biology, *optical* and *electronic*. Each could in principle be used *transmission mode* or *scattering mode*, and the image may be formed by *projection* onto a 2D sensor or by *scanning* a beam over the specimens.

Optical microscopes use light (usually visible or ultraviolet), almost always in transmission mode. The light that passes through the material is focused by a set of lenses, and projected onto a 2D sensor array which generates the digital image. Specimens can be observed in their natural state, as in figure 1.5 (left); or after a process of preparation and staining. The latter may involve flattening or cutting the organism into thin layers so that it becomes two-dimensional. Optical microscopes have a resolution limit of about 200 nm, since optical lenses cannot easily focus light with smaller wavelengths.

Optical images are usually acquired in color, with three or more separate color bands.

In this thesis, we only use monochromatic (gray scale) images; however the extension to color images is straightforward.

Electron microscopes use electrons instead of light. These particles have much shorter quantum wavelength than the photons of visible light, so they can form images with much greater resolution. In a *transmission-mode electron microscope* (TEM) the electrons that pass through the object are focused by magnetic “lenses” and projected onto a 2D sensor array. The fraction transmitted at each point of the object is represented in the image by a shade of gray. For this type of microscope, the material must be cut into very thin layers. See figure 1.5 (center).

Electron microscopes often use the scanning beam approach. In such a *scanning electron microscope* (SEM), the image is formed one pixel at time by a moving beam of electrons that sweeps over the specimen. Typically one measures the electrons scattered by the specimen instead of transmitted through it. SEM in the scattering mode can be used with specimens of arbitrary thickness, but it generally requires the object to be coated with a thin layer of metal, since uncoated organic substances do not scatter enough electrons to form a usable image. SEM images typically have a characteristic three-dimensional shadowed appearance, and show only the outer surface of the organism. See figure 1.5 (right).

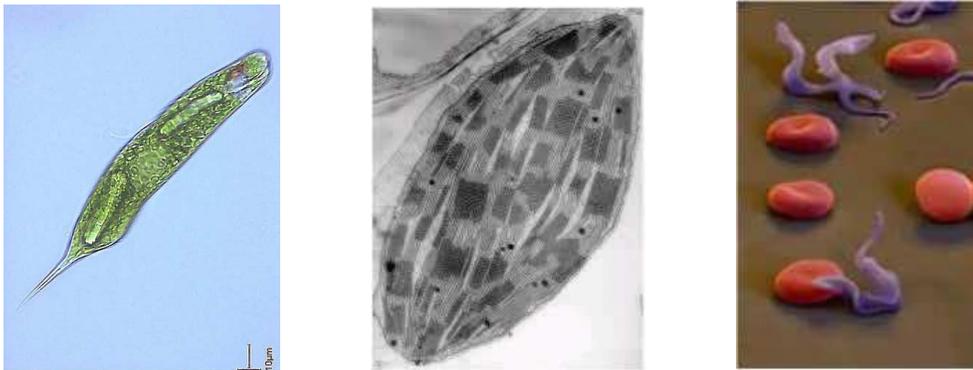


Figure 1.5: Images obtained by different types of microscopes [7]. Left: Optical transmission microscope. Center: Transmission electron microscope. Right: Scanning electron microscope.

In this project, we consider only instances of the problem where the relationship between the 3D object and its 2D image can be described by a geometric projection. That requirement means images of unprepared specimens obtained with optical microscopes, and images of coated specimens obtained with scanning electron microscopes. We exclude images of prepared or sectioned specimens (such as most TEM images), since they imply a

much more complicated correspondence between the 3D schematic model of the organism and the its 2D image.

## 1.3 Computational challenges

In order to solve the organism matching problem, one has to overcome several conceptual and computational challenges. First, one must use modeling techniques that can adequately reproduce the relevant features of the target organism and their deformations. One also must be able to render the organism model taking into account their transparency and the lighting effects; or alternatively, one has to develop ways to analyze the given images that are insensitive to these effects.

Most research on the recognition of deformable models considers a relatively small range of deformations and rotations of the model, characterized by a small number of parameters. For the organisms considered in this work, the models require a large number of parameters and the deformations may be quite extreme as seen in figure 1.3 (although the organism usually remains recognizable even in such cases). Therefore, to overcome this problem we had to use sophisticated multiscale methods of alignment and recognition.

### 1.3.1 The exponential cost barrier

In real situations, the size  $\rho$  of the possible range of each pose or deformation parameter is much larger than the required accuracy  $\delta$  in that parameter. A “brute force” solution would enumerate a grid of points in the parameter space with step  $\delta$ . However, the number of such points would be in the order of  $(\rho/\delta)^n$ , where  $n$  is the number of parameters. For typical values of  $\rho$ ,  $\delta$  and  $n$ , this number is astronomical.

## 1.4 Multiscale alignment strategy

The exponential cost barrier can be overcome by *multiscale matching* techniques [24, 58, 57, 59]. This approach uses two or more versions  $A^{(0)}, A^{(1)}, \dots, A^{(m)}$  of the entity  $A$  involved — in our case, the input digital image  $A$  — each with a different level of detail. The problem at hand is solved first for the coarsest scale version  $A^{(m)}$  yielding a tentative set of parameters. This solution is then used as an initial guess to solve the problem for the next scale  $A^{(m-1)}$ , where there is a bit more detail yielding an improved set of parameters. This process is repeated until reaching the original scale  $A^{(0)}$ . See figure 1.6. Since only small parameter adjustments need to be considered at each stage, the total running time is greatly reduced in comparison with the brute-force approach.

On the other hand, since the best match at some coarse scale may not correspond to the best match at scale 0, it is necessary to keep several candidates matches at each scale and reduce this set as the matching progresses to the finer scales.

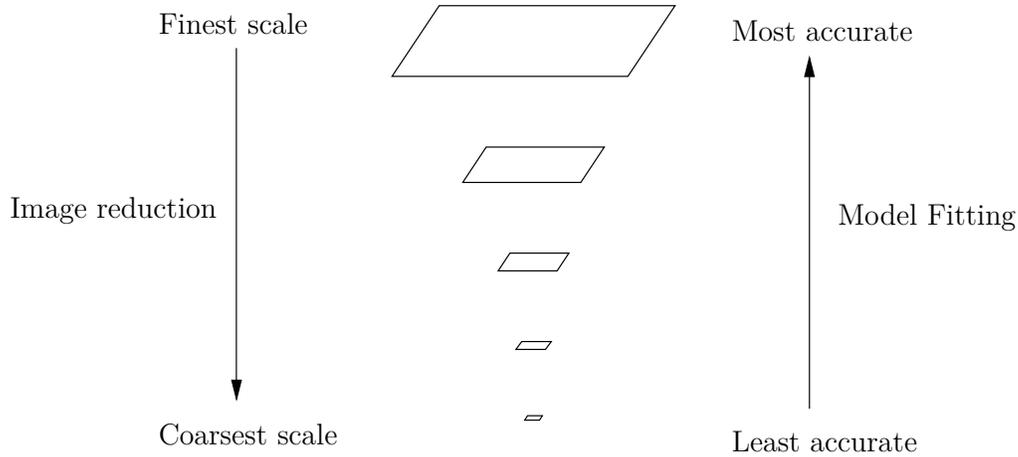


Figure 1.6: Schematic diagram of the multiscale 3D-2D alignment.

## 1.5 Organization of the thesis

The body of this thesis is organized as three parts and one appendices. Part I (Chapters 3 and 4) defines the basic notation and the image discrepancy metrics used by MSFIT. Part II (Chapters 5) describes the algorithm MSFIT in abstract terms and the details the specific variant we implemented. Part III (Chapters 6, 7 and 8) describes models and images used in our tests, the performance scores, and the results of several tests that are meant to justify some of the choices made in Part II. Appendix A describes the novel nonlinear optimization algorithm.

# Chapter 2

## Related work

### 2.1 Specific methods

Most previous work on this problem (determining pose and deformations of flexible objects in images) considers objects of very specific classes, such as human faces [16, 18], human bodies [21, 12, 35, 61], and teeth [56], and therefore develop fitting methods specialized for each class. Furthermore, many of these methods are specific to digital photos of macroscopic objects, and use features such as color and shading characteristics that are not reliably recognizable in our context.

Some of these methods use markers that are specific to each object or model. For example, in the case of faces, the markers may be obtained by identifying the eyes, nose and mouth [29, 60].

#### 2.1.1 Facial recognition

There is an extensive bibliography on fitting 3D face models to images. Volker and Vetter et al. [17] describe a method to fit 3D deformable face models to images. It can be seen as a generalization of the Active Appearance Model for fitting 2D objects images [23]. Their deformable model is linear combination of a dataset of annotated face models obtained by PCA, and a linear combination of texture maps applied to them. The fitting procedure requires an initial guess provide by user. The fitting optimizes a cost function that take into account texture. The optimization uses the multiscale approach, in which the fitting obtained with a low resolution image is used as the initial guess for a more detailed version of the same.

Chen and Wang [19] describe a method for fitting 3D model faces to images using 3D Active Appearance Model, that is quite similar to previous method; except that it uses the gradient of the input image to linearize cost function. It also includes an automatic

procedure to obtain the initial guess. Assen et al. [30] too described an application of 3D Active Appearance Model to align and segment the left ventricle of the heart from computed tomographic images.

Abboud et al. [11] fit a 3D face model to a single images using 3D Active Appearance Model and use the shape parameters of the fitting as input of machine learning method to classify facial expression.

Gao et al. [28] present a review of the Active Appearance Models and show some applications of this method, such as the facial expression recognition, tracking of deformable objects, fitting facial 3D models to images and medical image segmentation. This work presents two main dissantagens of this method, that are: 1) the accuracy are very dependent on the texture; 2) some applications still require manual landmarks.

To increase the fitting accuracy, some methods use more than one picture of the same object taken from different points [21, 31]. The method proposed by Chu and Nevatia [21], for example, fits a human body model to various independently images taken from different viewpoints. Hu et al. [31] present a theoretical background that allow to fit a single 3D Active Appearance Model to multiple images simultaneously, improving the fitting robustness.

### 2.1.2 Medical images

A survey of computer vision including a discussion about many methods for image segmentation and matching deformable models was published in 1996 by Terzopoulos et al. [40] mostly in the context of medical image analysis. It points out many difficulties that exist in medical image fitting and tracking, for example the lack of reliable landmark points in the smooth organs like the heart.

An example of this class is the work of Assen et al. [30] that fits a 3D Active Appearance Model of the left ventricle of the heart to computed tomographic images. Iwashita et al. [32] described an application that align a 3D rigid model of the gallbladder to video frames.

### 2.1.3 Hands

Recently much attention has been given to the recognition of hand gestures in images. However, many of the published methods use 2D hand models while others require a depth map (obtained by some stereo vision technique) to assist the fitting of the 3D model.

The method of Gorce et al. is an exception. It fits a 3D articulated skeleton model of the hand to sucessive video frames. The cost function used is the color difference between the input frame and the synthetic image.

### 2.1.4 Multiple image fitting

To increase the fitting accuracy, some methods use more than one picture of the same object taken from different points [22, 35]. The method proposed by Chu and Nevatia, for example, fits a human body model to various images taken from different viewpoints. Hu et al. give the theoretical background that allow to fit a single 3D Active Appearance Model to multiple images simultaneously, improving the fitting robustness.

## 2.2 Generic methods

### 2.2.1 Model-free methods

Falcão et al. [49] described methods for the recognition of microscopic parasites of arbitrary shape using image segmentation and various shape and texture invariants. Those methods have good performance but do not deliver the pose and deformation parameters of the organism.

### 2.2.2 Rigid methods

Many published methods can cope with generic objects, but only rigid ones such as cars, airplanes and mechanical parts subjected only to rigid translations and rotations. A typical example was described by Toshev et al. [55] for tracking solid objects without occlusion in a video. Their method precomputes a set of 500 projections for each model from many directions then tries to match their outlines to the outline of the object in each video frame, exploiting temporal coherence. This approach would be impractical with deformable models since it would need a prohibitively large number of precomputed images.

Another example is the method described by Yasuyo and Nobuyuki Kita [50]. Their method aligns a rigid model to two images. It uses the Canny operator to extract edges from the images, and then find a set of vertex (edge intersection) points, that are matched with the corresponding points of the projection of the model. Their algorithm is tolerant to occluded edges.

Minetto [41] is an example of another approach for locating and tracking rigid object, that uses a set of known fiducial marks attached to the object. Viola and Wells [51] too describe a method of alignment of rigid objects that uses the maximization of mutual information between the 3D model and the image.

## 2.3 Deformable models

Various of the methods presented in this chapter can be generalized but we did not find fully generic methods for deformable models.

Some of the methods listed in this section 2.1 can be generalized to the other deformable objects. Gao et al. [28] present a review of the Active Appearance Models and show some applications of this method, such as recognition of organs in medical images, faces and the other deformable objects. It points out some disadvantages of this method, such as the accuracy being very dependent on the texture and the need for manual identification of landmarks in some applications.

However, we did not find any fully general methods for fitting arbitrary deformable models to images. A fundamental limitation of most specific methods is that they assume a very limited range of deformations, and therefore can assume good initial guesses. Also, many methods (like AAM) are based on landmark points and lines that do not always exist on generic deformable models.

## 2.4 Tools

### 2.4.1 Metrics

Zampieri et al. [58] introduced the normalized gradient distance as a metric for image comparison that is largely insensitive to shading, illumination and rendering variations. They also introduced multiscale search, but limited to searching similar images with no changes in deformation, position, or pose.

### 2.4.2 Optimization methods

All these methods use various numerical minimization algorithms to find the optimal model parameters, including genetic search [39], particle swarm [36], Nelder-Mead simplex [42, 46], stochastic optimization [51] and others. However, we found that, in the multiscale approach, iterated quadratic optimization is more efficient and reliable, provided that the sampling points and the evaluation procedure are chosen so that the function has nearly quadratic behavior near the minimum. See section 4.4 and Appendix A.

### 2.4.3 Modeling techniques

The literature on the modeling of deformable biological structures is relatively modest [20, 40] and that on modeling of microorganisms is virtually non-existent. Anyway,

the adequate description of such structures and their deformations often requires ad hoc geometric modeling methods specific to each kind of organism.

### Modeling the shape

Our MSFIT algorithm can be used with a wide variety of geometric modeling techniques. In any case, the models must be able to represent the approximate rest shape of the organism and allow its normal deformations. For typical applications, the model need not be highly accurate or detailed, but must allow efficient rendering. Techniques that can be used for this purpose include as triangular meshes, Bézier patches, constructive solid geometry (CSG) formulas, blobs, etc. See figure 2.1.



Figure 2.1: Two simple geometrical models of a red cell, one using 8 Bézier patches (left), and another one using three blob elements whose centers are indicated by the dots (right). Both models were rendered with POV-Ray [8].

### Modeling the deformation

There is an extensive literature on modeling 2D *deformations* [47, 33, 15, 22], especially on the problem of aligning two-dimensional images taken with the same pose and view point — for example, images of consecutive sections of an organism. However these 2D techniques are of little use in our problem. The literature on modeling 3D deformations is more limited.

A straightforward solution is to vary the parameters of the geometric model itself, such as the position of the control points of Bézier patches, centroids and radii of blobs elements, etc. However this solution is very expensive since typical shape models (such as triangle or Bézier meshes) may have hundreds of independent parameters.

Another way to model the deformation is to immerse the geometric model within a coarse grid of deformable elements [26, 25], such as tricubic Bézier hexahedra. Each hexahedron can be understood as cube of elastic material, whose position and shape in space are controlled by a grid of  $4 \times 4 \times 4 = 64$  control points. Any deformation applied to this mesh is then interpreted as a deformation of 3D space, and transferred to the model which is immersed in it. This technique is very general since it does not depend on the kind or complexity of the geometric model; and can produce fairly smooth deformations. However, it can be quite expensive to render, since casting a straight ray in deformed space requires tracing a curved path in the original (undeformed) model.

A better solution is to choose a small set of *deformation modes*, where each mode consist of a simultaneous change in all the model's shape parameters, in various directions and magnitudes. The overall deformation is then defined by a small number of coefficients that specify the amount or strength of each deformation mode. This technique is usually applied to triangular meshes, but is easily adapted for other modeling techniques such as Bézier meshes and blobs.

**Part I**

**Fundamental concepts**



# Chapter 3

## Fundamental concepts

### 3.1 The organism model

Our matching algorithm requires a *tridimensional deformable geometric model* of the organism to be recognized, i. e., a computer description  $\mathcal{M}$  of its schematic morphology and its allowed variations and deformations. We assume that the model has a list of  $n$  *parameters*  $p[0], p[1], \dots, p[n-1]$  that determine the *pose* (position and orientation in space) of the model relative to the microscope’s camera, and its *deformation* relative to its “rest” shape (including changes of length and width, as well the folding, position and pose of its parts). We denote by  $\mathcal{M}(p)$  the instance of  $\mathcal{M}$  whose geometry and pose are defined by the parameter vector  $p$ . Note that the number  $n$  of parameters is specific to each model.

We assume that the parameter vector  $p$  of a model  $\mathcal{M}$  ranges over some *parameter domain*  $\mathcal{D} \subseteq \mathbb{R}^n$  that is a *box* of  $\mathbb{R}^n$ , that is a Cartesian product  $D[0], D[1], \dots, D[n-1]$  of intervals. An *interval* is a set of real numbers of form  $[a\_b] = \{x \in \mathbb{R} : a \leq x \leq b\}$  where  $a, b$  are double-precision floating-point values or  $\pm\infty$ . We use the notation  $I.lo$  and  $I.hi$  for the lower and upper endpoints of an interval  $I$ . We also use  $I.md$  for the interval’s *midpoint*  $(I.lo + I.hi)/2$ , and  $I.rd$  for its *half-width* or *radius*  $(I.hi - I.lo)/2$ .

### 3.2 Formal statement of the problem

The inputs of our algorithm are a monochromatic digital image  $A$ , a geometric model  $\mathcal{M}$  with  $n$  parameters, and the parameter domain  $\mathcal{D}$ . The algorithm must *align* that abstract 3D model with the microscope image; that is, it must find a parameter vector  $p$  that provides the best match between the expected projection of the instance  $\mathcal{M}(p)$  of the model and the expected projection of the organism on the camera sensor (the given

image  $A$ ). The output of the algorithm is that parameter vector.

### 3.3 Rendering a model

Our multiscale alignment algorithm requires the ability to render an instance  $\mathcal{M}(p)$  of the geometric model  $\mathcal{M}$  with a given parameter vector  $p$ , generating a synthetic image  $\mathcal{S}(\mathcal{M}, p, k)$  with the same size as the corresponding image  $A^{(k)}$  of the pyramid. For this purpose we implemented an *image synthesis module* based on the ray-tracing approach of computer graphics [52, 53].

For matching to optical microscope images, the organism usually must be modeled with transparent materials (ideally the rendering procedure should account for refraction and wave-related optical phenomena such as diffraction and interference, but we have not explored these aspects in our thesis).

For matching to scanning electron microscope images, we can consider the organism to be opaque. The physics of production scattering and detection of electrons in a scanning microscope are somewhat different from those of visible light; however, since there is little transmission, interference or diffraction, the simulation of SEM imaging is actually simpler, and largely compatible with the Newtonian particle model [43] assumed in conventional ray-tracing.

Our image synthesis module also generates a binary mask  $W = \mathcal{W}(\mathcal{M}, p, k)$  that defines the object's outline, which is used when comparing the images. As explained in Chapter 6, the image  $\mathcal{S}(\mathcal{M}, p, k)$  is anti-aliased by shooting several rays within each pixel and taking the average of the ray's colors as the pixel's color.

# Chapter 4

## Discrepancy functions

A *discrepancy function* is a function  $\text{dist}(\cdot, \cdot)$  that takes two images and returns a real number that expresses as how dissimilar they are. The choice of discrepancy function is closely linked to the image synthesis process, since the less sensitive the discrepancy metric is to the variations of illumination and diffraction effects, the less effort will be needed in the synthesis of microscope images to simulate these effects.

In our algorithm, the arguments of a discrepancy function are typically an actual microscope image  $A$  of the organism and synthetic image  $B = \mathcal{S}(\mathcal{M}, p, k)$  of its model. It is always assumed that  $A$  and  $B$  have the same domain  $\mathcal{E} \in \mathbb{R}^2$ . The discrepancy functions that we use have a third parameter  $W$  a *weight mask* with the same domain  $\mathcal{E}$ . The value of  $W$  at each pixel is a non-negative real number that is interpreted as the relative importance of that pixel in the discrepancy function.

The mask  $W$  is used to eliminate the influence of background clutter and to speed up the computation of the discrepancy. Typically the mask is computed from the model, and defines the region of  $\mathcal{E}$  covered by its projection with some extra margin all around.

### 4.1 Euclidean discrepancy

One of the discrepancy functions that we use is the *mean Euclidean distance* between two monochromatic images  $A, B$  *weighted* by the mask image  $W$ :

$$\text{dist}_2(A, B, W) = \sqrt{\frac{\sum_{z \in \mathcal{E}} W[z](A[z] - B[z])^2}{\sum_{z \in \mathcal{E}} W[z]}} \quad (4.1)$$

It is worth noting that the value of  $\text{dist}_2$  is always a real number in  $[0, 1]$ , provided that the image samples  $A[z]$  and  $B[z]$  are in  $[0, 1]$ . Note also that the sums need to be computed only over those pixels  $z$  where the mask  $W$  is nonzero.

In the implementation, it is more efficient to work with the squared discrepancy. Besides saving some square roots, the squared discrepancy has a quasi-quadratic behavior near its minimum, when it is considered a function of the parameter that define  $B$ . This feature which is essential for the minimization algorithm (Appendix A).

## 4.2 Normalized gradient

As noted in Chapter 1, the actual pixel values of a microscope image vary enormously depending on the preparation and illumination conditions. In order to reduce the effect of these variations, we preprocess the images so as to highlight the organism's outline in white and map any regions with uniform color to black. For this purpose we use the *normalized gradient* image operator defined as

$$\nabla I = \frac{|\mathbf{S}I|}{\sqrt{\mathbf{V}I + \eta^2}} \quad (4.2)$$

where  $\mathbf{S}$  is the Sobel gradient image operator with a  $3 \times 3$  weight window,  $\mathbf{V}$  is the local image variance operator using a  $5 \times 5$  weighted window, and  $\eta$  is the assumed standard deviation of the per-pixel imaging noise. The difference between this normalized gradient and the ordinary gradient image  $\mathbf{S}$  can be seen in figure 4.1.



Figure 4.1: An image  $I$  (left), its Sobel gradient image  $\mathbf{S}I$  (center) and its normalized gradient image  $\nabla I$  (right).

The *normalized gradient discrepancy* or *outline discrepancy* is then defined as the Euclidean discrepancy between the normalized gradients of the images  $A$  and  $B$ , that is

$$\text{dist}_{\nabla 2}(A, B, W) = \text{dist}_2(\nabla A, \nabla B, W) \quad (4.3)$$

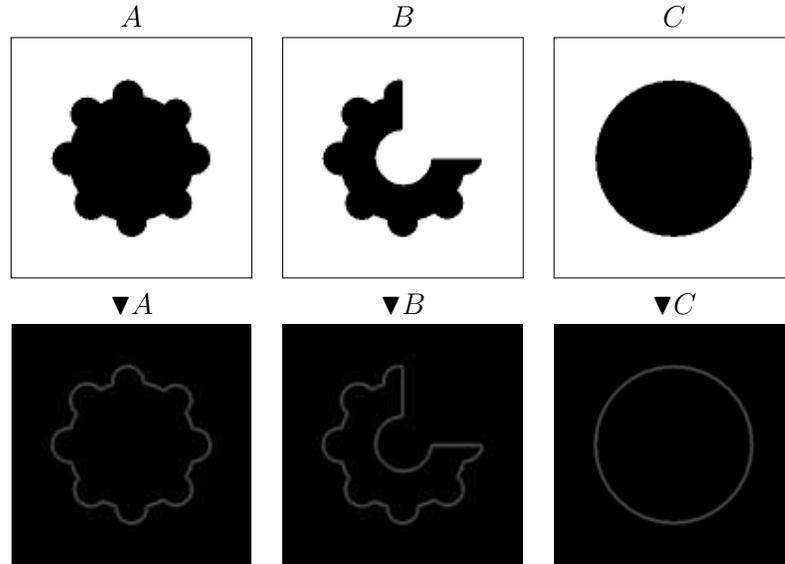


Figure 4.2: Images  $A$ ,  $B$  and  $C$  used in the comparison of discrepancy functions, (top) and their normalized gradient images  $\nabla A$ ,  $\nabla B$  and  $\nabla C$ .

#### 4.2.1 Comparison of $\text{dist}_2$ and $\text{dist}_{\nabla 2}$

To illustrate and compare the discrepancy functions  $\text{dist}_2$  and  $\text{dist}_{\nabla 2}$ , we use the three images  $A$ ,  $B$  and  $C$  shown in figure 4.2. The table 4.1 shows the discrepancies between  $A$  and the other two images using a uniform mask  $W$  (1 over the entire domain  $\mathcal{E}$ ).

Discrepancy function	$A : B$	$A : C$
$\text{dist}_2$	0.0943	<b>0.0548</b>
$\text{dist}_{\nabla 2}$	<b>0.0012</b>	0.0026

Table 4.1: Matching image  $A$  to images  $B$  and  $C$  using  $\text{dist}_2$  and  $\text{dist}_{\nabla 2}$ .

Note that image  $A$  is more similar to image  $C$  when using the discrepancy function  $\text{dist}_2$ , whereas with  $\text{dist}_{\nabla 2}$   $A$  is more similar to  $B$ .

### 4.3 Multiscale discrepancy

In the multiscale fitting algorithm, we typically use the chosen image discrepancy function  $\text{dist}(\cdot, \cdot)$  at a coarse scale  $k$  in order to estimate the discrepancy at the finest scale 0 (original). For that purpose, we define the shorthand

$$\text{dist}_2^{(k)}(A, B, W) = \text{dist}_2(A^{(k)}, B^{(k)}, W^{(k)}) \quad (4.4)$$

where  $A^{(k)}$  is usually the given image  $A$  reduced to scale  $k$ ,  $B^{(k)}$  is the synthetic image  $\mathcal{S}(\mathcal{M}, p, k)$  rendered at that scale from some parameter vector  $p$ , and  $W^{(k)}$  is the corresponding mask  $\mathcal{W}(\mathcal{M}, p, k)$  produced by the render.

In that context we also use the *cumulative multiscale discrepancy* that combines the discrepancies at some scale  $k$  and all coarser scales:

$$\text{dist}_2^{(\geq k)}(A, B, W) = \frac{\sum_{i=k}^m \alpha_i \text{dist}_2(A^{(i)}, B^{(i)}, W^{(i)})}{\sum_{i=k}^m \alpha_i} \quad (4.5)$$

where  $m$  is the coarsest level considered, and each  $\alpha_i$  is a fixed weight that defines the importance of scale  $i$  in the comparison. Typically, the weights  $\alpha_i$  are a geometric progression increasing with  $i$ , so that  $\text{dist}_2^{(\geq k)}$  does provide a rough estimate of  $\text{dist}_2^{(\geq 0)}$ .

### 4.3.1 Comparing single and multiscale discrepancies

Tables 4.2 and 4.3 show the multiscale discrepancies between the three images of the figure 4.2 under the discrepancies  $\text{dist}_2^{(\geq k)}(\cdot, \cdot)$  and  $\text{dist}_{\blacktriangledown 2}^{(\geq k)}(\cdot, \cdot)$ , respectively. Figures 4.3 and 4.4 show the respective images  $A^{(k)}$ ,  $B^{(k)}$ ,  $C^{(k)}$  and  $\blacktriangledown A^{(k)}$ ,  $\blacktriangledown B^{(k)}$ ,  $\blacktriangledown C^{(k)}$ . In all these comparisons we used the scale weights  $\alpha_k = 2^{k-m}$  and  $\alpha_k = 3^{-k}$ , with the image scale factor  $\mu = 1/2$ .

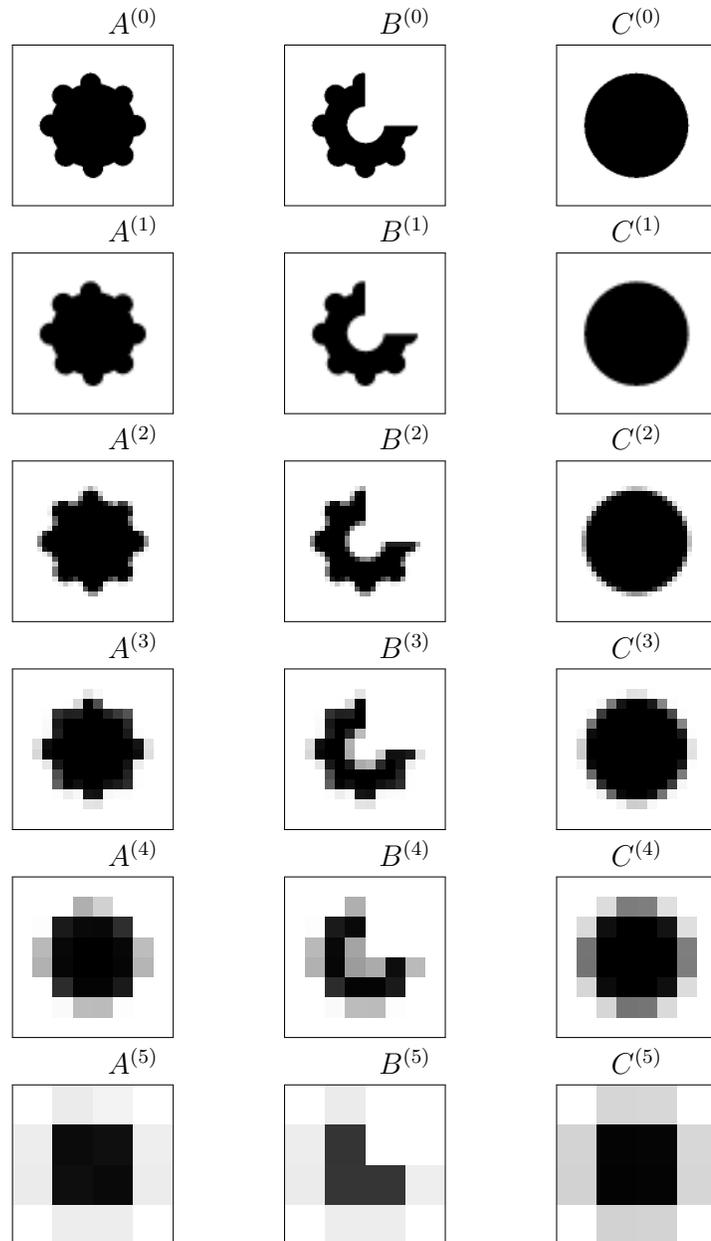


Figure 4.3: Reduced versions  $A^{(k)}$ ,  $B^{(k)}$  and  $C^{(k)}$  for the images of the figure 4.2. At scale 0 the images have  $128 \times 128$  pixels.

Discrepancy function	$k$	$\alpha_k$	$A : B$	$A : C$
$\text{dist}_2^{(k)}$	5		0.0602	<b>0.0051</b>
	4		0.0760	<b>0.0113</b>
	3		0.0845	<b>0.0345</b>
	2		0.0892	<b>0.0429</b>
	1		0.0927	<b>0.0499</b>
	0		0.0943	<b>0.0548</b>
$\text{dist}_2^{(\geq k)}$	5	1.0000	0.0602	<b>0.0051</b>
	4	0.5000	0.0655	<b>0.0072</b>
	3	0.2500	0.0682	<b>0.0111</b>
	2	0.1250	0.0696	<b>0.0132</b>
	1	0.0625	0.0703	<b>0.0144</b>
	0	0.0312	0.0707	<b>0.0150</b>
$\text{dist}_2^{(\geq k)}$	5	0.0041	0.0602	<b>0.0051</b>
	4	0.0123	0.0720	<b>0.0097</b>
	3	0.0370	0.0807	<b>0.0269</b>
	2	0.1111	0.0864	<b>0.0377</b>
	1	0.3333	0.0906	<b>0.0458</b>
	0	1.0000	0.0931	<b>0.0518</b>

Table 4.2: Matching image  $A$  to images  $B$  and  $C$  using  $\text{dist}_2^{(k)}$  and  $\text{dist}_2^{(\geq k)}$  with increasing and decreasing weights  $\alpha_k$ .

As in section 4.2.1, observe that  $A$  is more similar to  $C$  than  $B$  at all scales. Note also that, with  $\alpha_k = 2^{k-m}$  the value of  $\text{dist}_2^{(\geq 3)}$  is closer to  $\text{dist}_2^{(\geq 0)}$  than  $\text{dist}_2^{(3)}$  is to  $\text{dist}_2^{(0)}$ . On the other hand with  $\alpha_k = 3^{-k}$  the predictor  $\text{dist}_2^{(\geq 3)}$  is just as bad as  $\text{dist}_2^{(3)}$ .

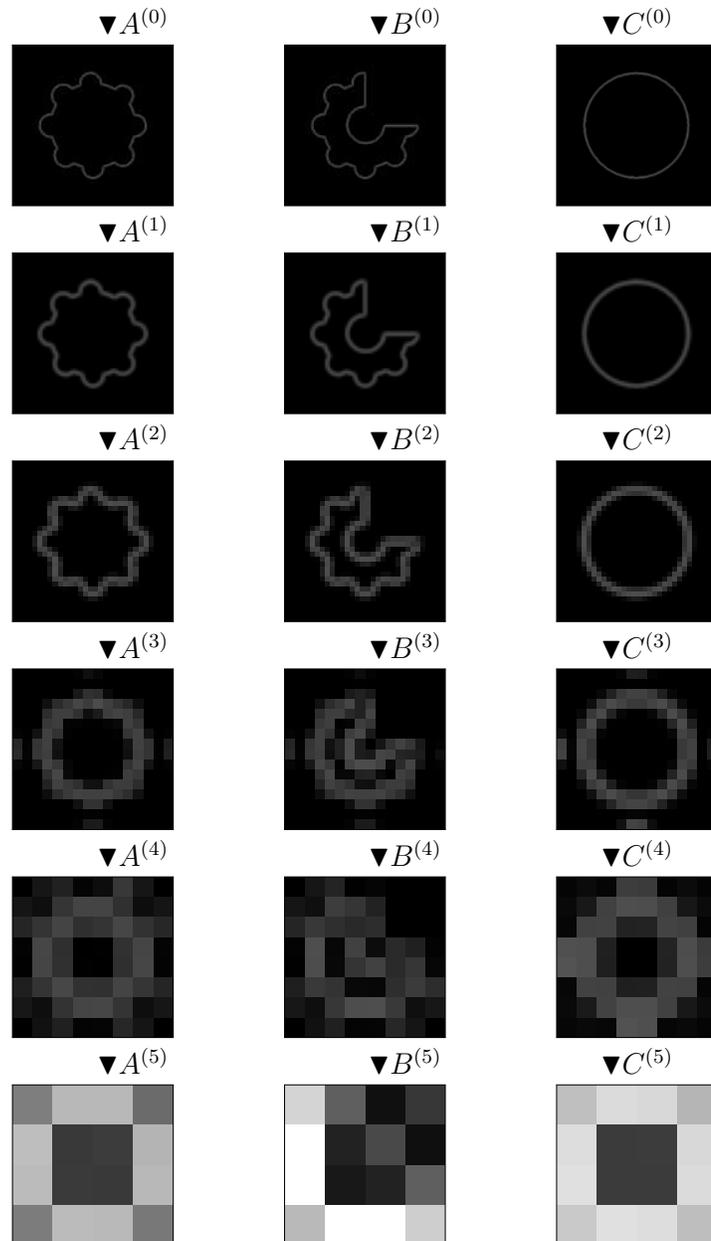


Figure 4.4: Normalized gradients  $\nabla A^{(k)}$ ,  $\nabla B^{(k)}$  and  $\nabla C^{(k)}$  of the images of figure 4.3.

Discrepancy function	$k$	$\alpha_k$	$A : B$	$A : C$
$\text{dist}_{\sqrt{2}}^{(k)}$	5		0.1079	<b>0.0288</b>
	4		<b>0.0091</b>	0.0121
	3		0.0067	<b>0.0038</b>
	2		<b>0.0045</b>	0.0056
	1		<b>0.0024</b>	0.0043
	0		<b>0.0012</b>	0.0026
$\text{dist}_{\sqrt{2}}^{(\geq k)}$	5	1.0000	0.1079	<b>0.0288</b>
	4	0.5000	0.0750	<b>0.0232</b>
	3	0.2500	0.0653	<b>0.0204</b>
	2	0.1250	0.0612	<b>0.0195</b>
	1	0.0625	0.0593	<b>0.0190</b>
	0	0.0312	0.0584	<b>0.0187</b>
$\text{dist}_{\sqrt{2}}^{(\geq k)}$	5	0.0041	0.1079	<b>0.0288</b>
	4	0.0123	0.0338	<b>0.0163</b>
	3	0.0370	0.0151	<b>0.0076</b>
	2	0.1111	0.0079	<b>0.0063</b>
	1	0.3333	<b>0.0043</b>	0.0050
	0	1.0000	<b>0.0022</b>	0.0034

Table 4.3: Matching image  $A$  to images  $B$  and  $C$  using  $\text{dist}_{\sqrt{2}}^{(k)}$  and  $\text{dist}_{\sqrt{2}}^{(\geq k)}$  with increasing and decreasing weights  $\alpha_k$ .

Observe that, with  $\text{dist}_{\sqrt{2}}^{(\geq k)}$ , image  $A$  is more similar to  $B$  or to  $C$  depending on the scale  $k$ . This variation reflects the fact that different details of the outline are prominent at each scale. Note that, with top-heavy weights  $\alpha_k = 2^{k-m}$  the discrepancies  $\text{dist}_{\sqrt{2}}^{(\geq k)}$  are more consistent, and that  $\text{dist}_{\sqrt{2}}^{(\geq 3)}$  is a fair estimate of  $\text{dist}_{\sqrt{2}}^{(\geq 0)}$ . The price for this consistency is that  $\text{dist}_{\sqrt{2}}^{(\geq k)}$  is dominated by the discrepancies at the roughest scales. On other hand, with bottom-heavy weights  $\alpha_k = 3^{-k}$ , we get an outline-sensitive discrepancy but  $\text{dist}_{\sqrt{2}}^{(\geq k)}$  is almost as irregular as  $\text{dist}_{\sqrt{2}}^{(k)}$ .

## 4.4 Effectiveness of the discrepancy functions

In this section we investigate the effectiveness of the discrepancy functions for the detection of correct parameter values. For this purpose we compare synthetic images  $A = \mathcal{S}(\mathcal{M}, \bar{p}(t^*), 0)$  and  $B = \mathcal{S}(\mathcal{M}, \bar{p}(t), 0)$  for some simple models  $\mathcal{M}$ , where the vector  $\bar{p}(t)$  varies along a straight line segment in  $\mathbb{R}^n$  depending on some real argument  $t$ , and  $t^*$  is the midpoint of that argument's range. Ideally, the plot of  $[\text{dist}(A, B, W)]^2$ , as a function of  $t$ , should have a single quadratic-like minimum (zero) at  $t = t^*$ .

In all tests of this section, the images have  $128 \times 128$  pixels at scale 0 and were produced by our ray tracer with  $10 \times 10$  rays per pixel. For the multiscale discrepancies, both the  $A$  and  $B$  images were rendered at each scale  $k$  (instead of being reduced from the scale 0 version), and we use the fixed weight  $\alpha_k = 3^{-k}$ .

#### 4.4.1 Moving disk

For the simplest test, which we will call **DISK**, we used a model whose projection is a black disk on a white background. The model depends on a single parameter  $X$  which is the horizontal position of its center  $(X,0)$ . The disk has a fixed radius of 16 pixels at scale 0. See figure 4.5.

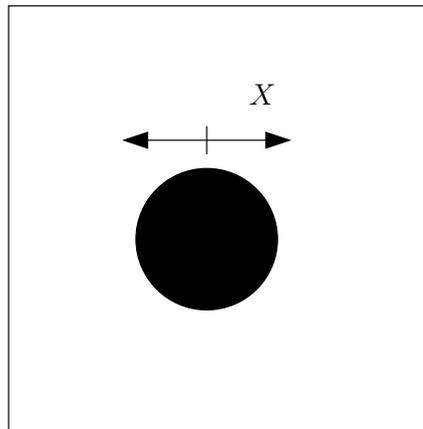


Figure 4.5: A typical image of the model used in test **DISK**.

Figure 4.6 shows the discrepancies between the image  $A$  obtained with  $X = 0$  and various images  $B$  obtained with  $X$  varying from  $-2$  to  $+2$  pixels with increments of  $0.2$  pixels. Note that this increment is twice the spacing between the traced rays at scale 0 ( $0.1$  pixels in each axis).

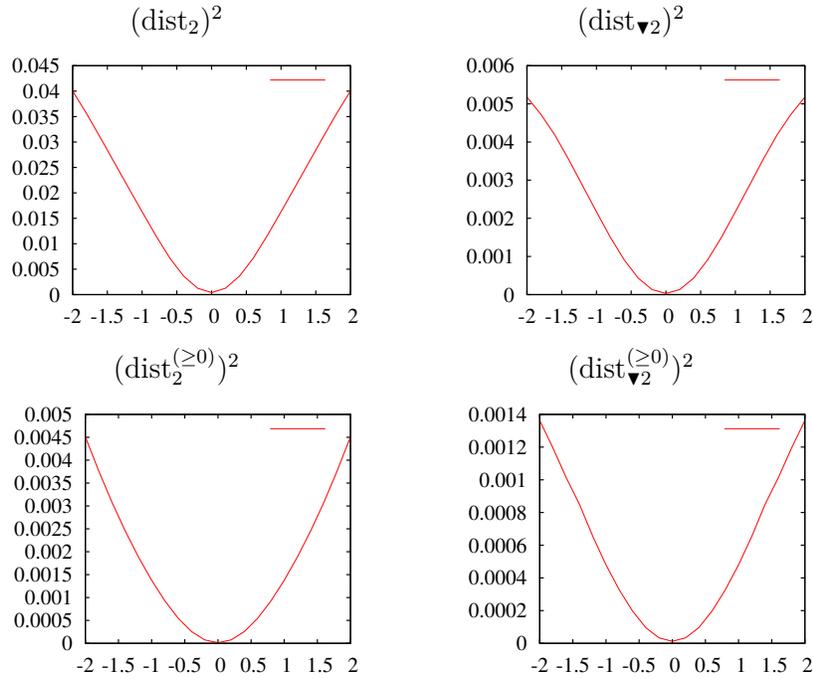


Figure 4.6: Behavior of squared discrepancy functions for a moving a disk as function of the horizontal displacement  $X$  (in pixels).

#### 4.4.2 Rotating rectangle

For the test `RECT`, we used a model whose image is a rectangle with fixed size (50 pixels by 20 pixels at scale 0) with fixed center  $(0,0)$  and rotated by an angle  $\theta$  about its center. See figure 4.7.

Figure 4.8 shows the discrepancies between the image  $A$  obtained with  $\theta = \theta_0 = 0.1$  radians and various images  $B$  obtained with  $\theta$  varying from  $\theta_0 - 2/27$  to  $\theta_0 + 2/27$  radians (corresponding to approximate a displacement of  $\pm 2/27\sqrt{25^2 + 10^2} \approx \pm 2$  pixels at the corners of the rectangle) with increments of  $1/135$  radians (corresponding to a displacement of approximately 0.2 pixels of those corners).

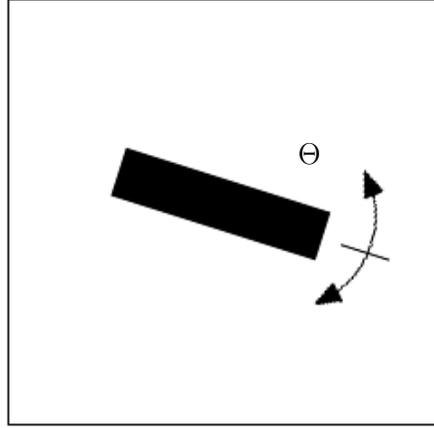
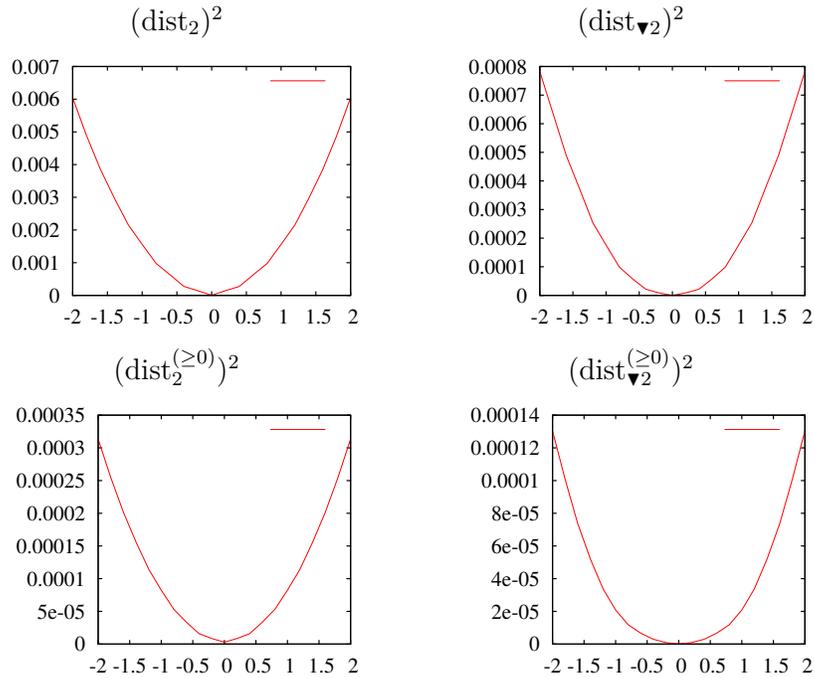


Figure 4.7: Image of the model used in test RECT, the rotating rectangle.

Figure 4.8: Behavior of the squared discrepancy functions for rotating rectangle example, as a function of the rotation angle  $\theta$ . The horizontal scale is the displacement of the corners, in pixels.

### 4.4.3 Bending rectangle

For test **BEND**, we used a model whose image is a rectangle with fixed size (98 pixels by 20 pixels at scale 0) with fixed center (0,0) which is bent by an angle  $\theta$  about its center. The rectangle is actually modeled by a Bézier patch primitive (see Chapter 6) whose control points are displaced so as to approximate the effect of bending (as in the **Eleg** model, section 6). Figure 4.9 shows the discrepancies between the image  $A$  obtained with  $\theta = \theta_0 = 0.1$  and various images  $B$  obtained with  $\theta$  varying from  $\theta_0 - 2/52$  to  $\theta_0 + 2/52$  radians (corresponding to a displacement of approximately  $\pm 2$  pixels at the corners of the rectangle) with increments of  $1/260$  radians (corresponding to a displacement of approximately 0.2 pixels)

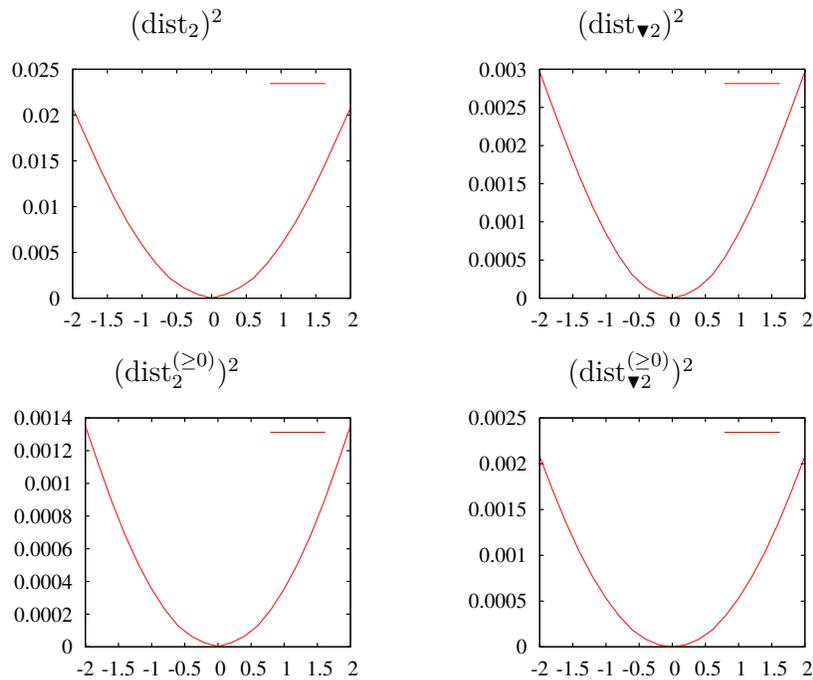


Figure 4.9: Behavior of the squared discrepancy functions for a bending rectangle, as a function of the rotation angle  $\theta$ . The horizontal scale is the displacement of the corners, in pixels.

### 4.4.4 Conclusion

These tests show that the squares  $\text{dist}_2$  of all discrepancy functions considered have smooth quadratic behaviour, near the minimum.

## 4.5 Granularity of discrepancy functions

It should be noted that the ray tracing method used to produce the synthetic images is discontinuous, since the value of each pixel changes suddenly whenever a ray starts hitting or missing the model. Therefore, the discrepancy function can be considered smooth only if the shape changes between renderings are greater than spacing between rays. An example of this effect can be seen in 4.10. Both graphs show the results of **RECT** test when the images are rendered with different number of rays per pixel. In these graphs the angle  $\theta$  was varied in increment of  $1/540$  radians (corresponding to a displacement of  $\approx 0.05$  pixels). The graph of right shows granularity in the discrepancy function due to ray hit/miss events. This step-like behaviour is very problematic for the nonlinear optimization algorithm. Care must be taken so that the spacing between samples of the goal function is never smaller than the width of those steps.

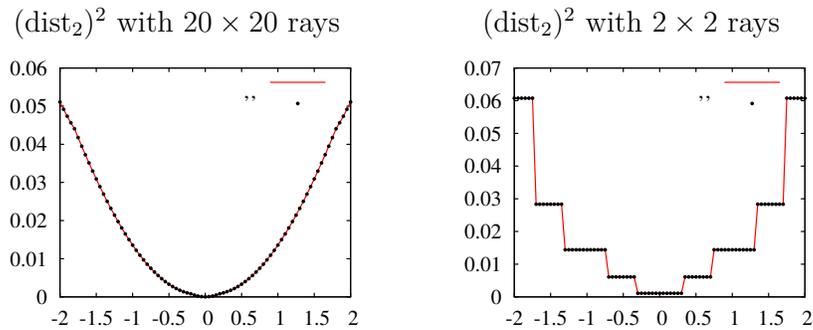


Figure 4.10: Plots of the squared discrepancy  $(\text{dist}_2)^2$  for the **RECT** example, with different number of rays per pixel. The black dots represent the sampled values.



**Part II**

**Multiscale Fitting**



# Chapter 5

## MSFit algorithm

### 5.1 Simplified algorithm description

A high-level description of our multiscale model fitting heuristic is the procedure MSFIT below.

---

**Procedure 1** MSFIT( $A, \mathcal{M}, \text{dist}, t$ )

---

**Requires:** An input image  $A$  of an organism, a deformable model  $\mathcal{M}$  of the same, an image discrepancy function  $\text{dist}$  and the desired number  $t$  of solutions.

**Returns:** A list of  $t$  parameter vectors  $p_1, p_2, \dots, p_t$  presumably corresponding to the  $t$  best fittings of  $\mathcal{M}$  to  $A$ .

- 1:  $A^{(0)}, A^{(1)}, \dots, A^{(m)} \leftarrow \text{BUILDPYRAMID}(A, \mathcal{M})$ ;
  - 2:  $\mathcal{Q} \leftarrow \text{INITIALCANDIDATES}(A^{(m)}, \mathcal{M}, m)$ ;
  - 3: **while** NOT ADEQUATE( $\mathcal{Q}$ ) **do**
  - 4:    $c \leftarrow \text{REMOVECANDIDATE}(\mathcal{Q})$ ;
  - 5:    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \text{CHILDREN}(c, \mathcal{M})$ ;
  - 6:    $\mathcal{Q} \leftarrow \text{TRIM}(\mathcal{Q})$ ;
  - 7: **end while**
  - 8: **return** SOLUTION( $\mathcal{Q}$ );
- 

The MSFIT procedure begins by creating a pyramid of images  $A^{(0)}, A^{(1)}, \dots, A^{(m)}$  (steps 1 and 2). The coarsest viable level  $m$  of the image pyramid is such that the model's projection, for any parameter vector  $p \in c.R$ , is expected to cover only one or two pixels.

The main state variable of the MSFIT is a *queue of candidates*  $\mathcal{Q}$ . Each candidate  $c$  is a record containing a vector  $c.R$  of  $n$  *parameter ranges*  $c.R[0], c.R[1], \dots, c.R[n-1]$ , each  $R[i]$  being an interval of allowed values for parameters  $p[i]$ ; and an integer *scale*  $c.s$  that indicates the level of the pyramid of images where candidate  $c$  has been tentatively identified. The vector  $c.R$  can be viewed as an  $n$ -dimensional box  $c.R[0] \times c.R[1] \times \dots \times$

$c.R[n - 1]$ , called the *total parameter range* of the candidate  $c$ .

In step 3, a set of initial candidates is created for scale  $m$ . At that level, any salient spot on the image with one or two pixels must be considered a possible instance of the target organism.

At each iteration of the main loop (steps 3 to 7), MSFIT removes a candidate  $c$  from the queue  $\mathcal{Q}$ , chosen by some criterion (step 4). The candidate  $c$  is split into zero or more *children* candidates  $c'$  with smaller and non-overlapping parameter ranges contained in  $c.R$ , that are inserted back into the queue  $\mathcal{Q}$  (step 6). We say that  $c$  is the *parent* of these children candidates. Then excess and/or non-viable candidates are removed from  $\mathcal{Q}$  (step 7). Steps 4 to 7 are repeated until the  $t$  desired solutions are found among the candidates in  $\mathcal{Q}$  (step 3). The parameter vectors of those  $t$  candidates are then returned (step 8).

The various sub-procedures used in MSFIT will be detailed in the next chapter. Each of them can be implemented in various ways. Different choices result in various specific versions of the algorithm, differing in robustness, performance and other aspects. The main queue management procedures (ADEQUATE, REMOVECANDIDATE and TRIM) strongly affect the growth of the queue  $\mathcal{Q}$ , the total execution time, and the expected quality of the results.

### 5.1.1 Candidate level

The attribute  $c.s$  of a candidate  $c$  specifies the level of the image pyramid at which the candidate is to be evaluated. More precisely, the merit of the candidate for selection (step 4) or trimming (step 6) is evaluated by comparing the image  $A^{(c.s)}$  with the synthetic image  $\mathcal{S}(\mathcal{M}, p, c.s)$  for some parameter vector  $p$  in the box  $c.R$ .

Each candidate  $c$  in the queue satisfies the following *range invariant*: as the parameter vector  $p$  ranges in the box  $c.R$ , the position of any point of the model on the synthetic image  $\mathcal{S}(\mathcal{M}, p, c.s)$  does not change by more than half pixel from its position on the “mean” image  $\mathcal{S}(\mathcal{M}, c.R.md, c.s)$ .

### 5.1.2 Active parameters

At each resolution level, there may be some parameters whose effect is not observable and therefore cannot be estimated in the image at that level. Namely, the range invariant may be satisfied even if some parameter range  $c.R[i]$  is the whole range  $D[i]$  of that parameter. We then say that the parameter  $p[i]$  is *inactive* (irrelevant) at that scale. For instance, at a scale where the organism’s projection is at most one or two pixels wide, only its position and approximate size can be meaningfully estimated from the image; all other parameters will be inactive. The rotation parameters are active only at scales where the object is large enough for its overall shape to be distinguished from a sphere. Parameters

that affect only the position of small appendages or organelles are active (meaningful and relevant) only at levels where these details are visible.

In step 6, the algorithm must choose the parameters ranges of the children of each candidate, so as to satisfy the range invariant. The children’s range  $c'.R[i]$  of a parameter  $p[i]$  typically depends on the scale  $c.s$  and on ranges  $c.R[j]$  of other parameters in a model-dependent way. On the other hand the procedure should avoid creating more children than necessary. These criteria automatically cause each parameter to be introduced in the search (that is, to become active) at the scale where it begins to be relevant.

Note that a parameter  $p[i]$  may be active or not for a candidate  $c$ , even at the same scale, depending on the range  $c.R[j]$  of other parameters. For instance, the rotation parameters may be active if the range of the size parameters includes large enough values; and will be inactive if the size parameters are limited to sufficiently small values.

## 5.2 Detailed algorithm

In this section we describe a specific version of the basic MSFIT algorithm which is relatively fast and easy of implement, but is not deterministic and does not always returns the best result.

In this approach, besides the vector parameters range  $c.R$  and level  $c.s$ , each candidate also has a vector of *nominal parameters*  $c.p \in c.R$  and a *nominal discrepancy* value  $c.d$ . Typically,  $c.p$  is an estimate for the “best” parameter vector in  $c.R$ , and  $c.d$  is obtained from the discrepancy  $\text{dist}(A^{(c.s)}, \mathcal{S}(\mathcal{M}, c.p, c.s), \mathcal{W}(\mathcal{M}, c.p, c.s))$  between the image  $A$  and the model  $\mathcal{M}$  at scale  $c.s$ .

In this heuristic, the REMOVECANDIDATE procedure tries to choose the “most promising” candidate, the one which has the greater chances of leading to a desired optimum match. Specifically, it chooses a candidate  $c$  from the queue  $\mathcal{Q}$  with maximum scale  $c.s$  and minimum discrepancy  $c.d$  among all candidates of that scale. With this policy, at any iteration all candidates in  $\mathcal{Q}$  have at most two different levels,  $k$  and  $k - 1$ . The algorithm splits all candidates of the higher level  $k$  before considering any candidate of the level  $k - 1$ .

The children candidates created by the CHILDREN have ranges  $c'.R$  whose union may be smaller than the parent’s range  $c.R$ . Specifically, the range  $c.R$  is first partitioned into children ranges  $R_1, R_2, \dots, R_r$ , and then a non-linear optimization [44, 27] is used on each sub-range  $R_j$  to determine the nominal parameter vector  $c'.p$  of each child and a restricted range  $c'.R \subseteq R_j$ . In our MSFIT implementation we use an original method for nonlinear optimization that is described in Appendix A.

The TRIM procedure keeps, for each level  $k$ , only the best  $L_k$  candidates with level  $c.s = k$  (in the sense of having the smallest estimates  $c.d$ ), and discards the rest; where  $L_k$

depends on the level  $k$  and the number of desired results  $t$ . In particular,  $l_0$  is  $t$ . Therefore, the maximum number of candidates stored in  $\mathcal{Q}$  will be  $L_0 + L_1 + \dots + L_m$ , and the maximum number of children candidates generated in step 6 will be  $L_m + r_{\max}(L_1 + \dots + L_m)$ , where  $r_{\max}$  is the maximum number of children of any candidate. After extensive testing we found that setting  $L_k = \lceil \alpha k \rceil + t$  for some constant  $\alpha = 0.5$  provides an adequate balance between speed and accuracy.

In this version of the algorithm, the ADEQUATE procedure returns true if and only if every candidate  $c$  in  $\mathcal{Q}$  has level  $c.s = 0$ . At that moment, the queue  $\mathcal{Q}$  will have exactly  $t$  candidates, and the SOLUTION procedure simply returns those candidates.

### 5.2.1 Hereditary discrepancy

Ideally, the nominal discrepancy  $c.d$  should be either the discrepancy between the given image  $A$  and the image of the model  $\mathcal{M}$  rendered with parameters  $c.p$  at scale  $c.s$ , or the multiscale version of that discrepancy. That is, we should have  $c.d = D_S(c, A, \mathcal{M})$ , where

$$D_S(c, A, \mathcal{M}) = \text{dist}(A^{(c.s)}, B^{(c.s)}(c.p), W^{(c.s)}(c.p)) \quad (5.1)$$

in the single scale version, and

$$D_M(c, A, \mathcal{M}) = \frac{\sum_{i=c.s}^m \alpha_i \text{dist}(A^{(i)}, B^{(i)}(c.p), W^{(i)}(c.p))}{\sum_{i=c.s}^m \alpha_i} \quad (5.2)$$

in the multiscale version, where  $B^{(k)}(p) = \mathcal{S}(\mathcal{M}, p, 0)$  and  $W^{(k)}(p) = \mathcal{W}(\mathcal{M}, p, 0)$  both reduced to level  $k$ .

In either case, however, computing  $\mathcal{S}(\mathcal{M}, p, 0)$  and  $\mathcal{W}(\mathcal{M}, p, 0)$  for every candidate would be too expensive. Therefore in formulas (5.1) and (5.2) we render the model at level  $k$ , that is we use  $B^{(k)}(p) = \mathcal{S}(\mathcal{M}, p, k)$  and  $W^{(k)}(p) = \mathcal{W}(\mathcal{M}, p, k)$ . We justify this optimization by noting that, with a well designed model and a good rendering procedure, the image rendered at level  $k$  should be very close to the image rendered at scale 0 and reduced to level  $k$ .

Even with this optimization, the computation of the discrepancy  $\text{dist}(A^{(k)}, \mathcal{S}(\mathcal{M}, c.p, k), \mathcal{W}(\mathcal{M}, c.p, k))$  for all levels  $k \geq c.s$  in the formula (5.2) would be about  $1/(1 - \mu^2)$  times as expensive as the computation of the discrepancy for level  $k$  only. Therefore, in equation (5.2) we instead assume that the discrepancies at higher levels do not vary much when  $p$  varies within its range  $c.R$ . Therefore we replace the sum (5.2) by

$$D_H^{(k)}(c, A, \mathcal{M}) = \lambda_k \text{dist}(A^{(k)}, B^{(k)}(c.p), W^{(k)}(c.p)) + (1 - \lambda_k)c.h; \quad (5.3)$$

Here  $c.h$  is an additional field of candidate  $c$  that stores the discrepancy  $c''.d$  inherited

from its parent  $c''$  (or 0 if  $c.s = m$ ), and  $\lambda_k = \alpha_k / \sum_{i=k}^m \alpha_i$ . The field  $c.h$  is set when the candidate  $c$  is created from its parent by the CHILDREN procedure.



## Part III

### Models and tests



# Chapter 6

## Modeling technique

### 6.1 Model instantiation

In our implementation, a deformable model  $\mathcal{M}$  is represented internally as an object with several internal data fields and methods. The most important method is `BUILDGEOMETRY`( $n, p, \mu, k$ ), which builds a fixed geometric description of the organism, given its parameters  $p[0], p[1], \dots, p[n-1]$ , suitable for rendering at level  $k$  of the image pyramid.

The description returned by `BUILDGEOMETRY` is a list  $G$  of geometric primitives whose union defines the organism's model shape. The geometric primitives currently implemented in our ray-tracer are described below and can be seen in figure 6.1. Additional primitives and other CSG operations [34, 13] (such as intersection and difference) could easily be added to the code.

- `SPHERE` - a sphere with center  $(0, 0, 0)$  and radius 1.
- `CUBE` - a cube with center  $(0, 0, 0)$  and side  $2/\sqrt{3}$  (i. e. radius 1).
- `CYLINDER` - a cylinder parallel to the  $z$ -axis with center  $(0, 0, 0)$ , base radius 1 and height 1.
- `BEZIER`( $b_{00}, b_{01}, b_{02}, b_{03}, b_{10}, b_{11}, \dots, b_{33}$ ) is a bi-cubic tensorial surface patch [37] defined by the 16 Bézier control points  $b_{00}, \dots, b_{33}$ .

Every instance of a primitive shape in the geometric description  $G$  also has an associated  $3 \times 3$  *transformation matrix*  $M$  and a *translation vector*  $t \in \mathbb{R}^3$  that should be applied to the basic primitive shape. Specifically a point  $q$  of the primitive (represented as column 3-vector) corresponds to the point  $Mq+t$  of its instance. Finally each instance of a primitive shape has a monochromatic optical coefficients (diffusion, ambient and transmission), similar to POV-Ray.

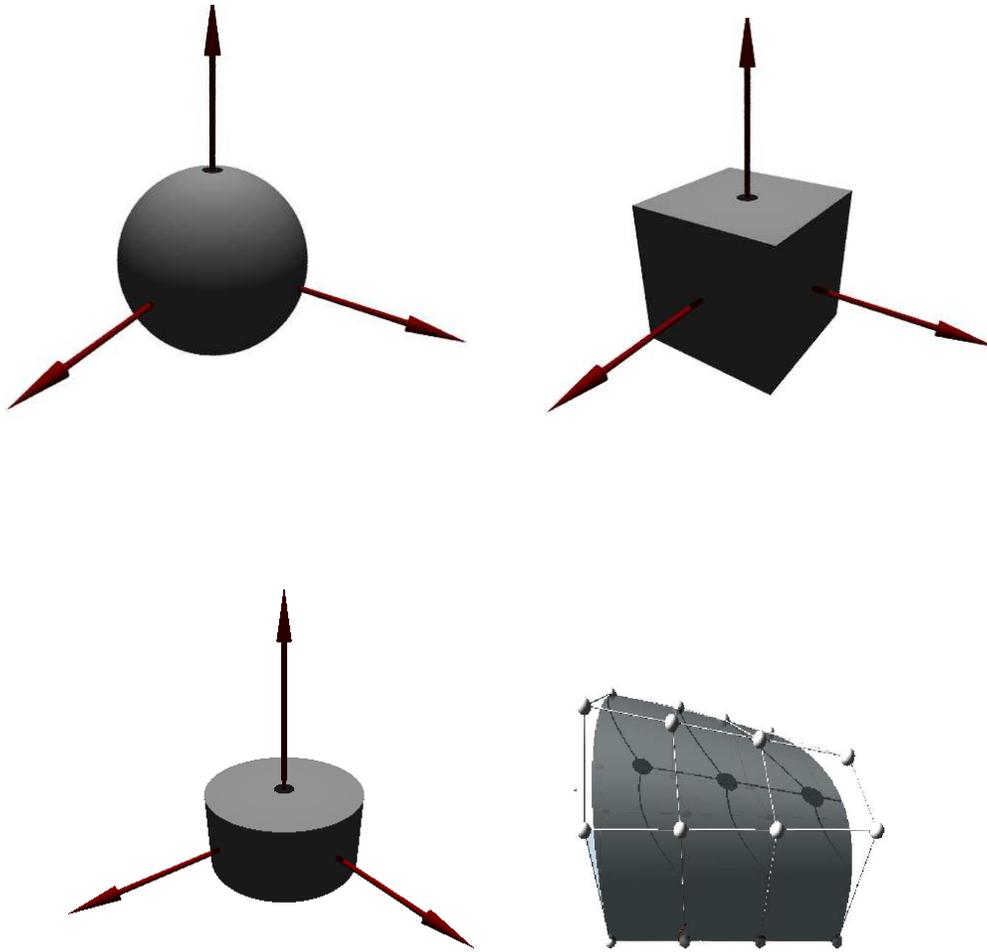


Figure 6.1: Geometric primitives: SPHERE, CUBE, CYLINDER and BEZIER respectively. The BEZIER illustration shows the 16 control points.

The BUILDGEOMETRY procedure converts the parameters  $p[0], p[1], \dots, p[n-1]$  to the primitive parameters (if any) and the modifiers  $M$  and  $t$ . By convention, all coordinates in the attributes of these instances of primitives are measured in pixels of the level 0 image.

For some models, the BUILDGEOMETRY procedure could generate very different descriptions depending on the parameter vector  $p$  and scale  $k$ . For example, if the diameter of the organism as described by  $p$  is smaller than  $\mu^k$  (that is, one pixel at scale  $k$ ) the geometric description could be just a single SPHERE primitive. More generally, BUILDGEOMETRY may omit details of the geometric description if they are too small to be seen at the given scale.

Some models also have one or more *meta parameter*, that can chosen by the user and are fixed during execution of the MSFIT. These meta parameter can be used to generate variants of the organism.

## 6.2 Rendering procedure

The geometric description  $G$  returned by BUILDGEOMETRY is converted into an image by the rendering procedure RENDER. It has six parameters  $(G, n_x, n_y, k, \mu, n_r)$  where  $n_x$  and  $n_y$  determine the size (in pixels) of the image to be generated at scale 0,  $k$  is the scale of the desired image,  $\mu$  is the scale factor and  $n_r$  specifies number of the rays to cast per pixel ( $n_r \times n_r$ ). The output is the rendered image  $B^{(k)} = \mathcal{S}(\mathcal{M}, p, k)$  at scale  $k$  and its corresponding mask  $W^{(k)} = \mathcal{W}(\mathcal{M}, p, k)$ .

### 6.2.1 Scale of image

The renderer assumes that each pixel of the image at scale 0 is a square of unit side on the  $xy$  plane. If image  $B^{(0)}$  has  $n_x \times n_y$  pixels, at scale  $k$  the output of our ray-tracing is an image  $B^{(k)}$  with  $\lceil n_x \mu^k \rceil \times \lceil n_y \mu^k \rceil$  pixels, and each pixel of that image represents a square with size  $1/\mu^k \times 1/\mu^k$  pixels in the original image. See figure 6.2.

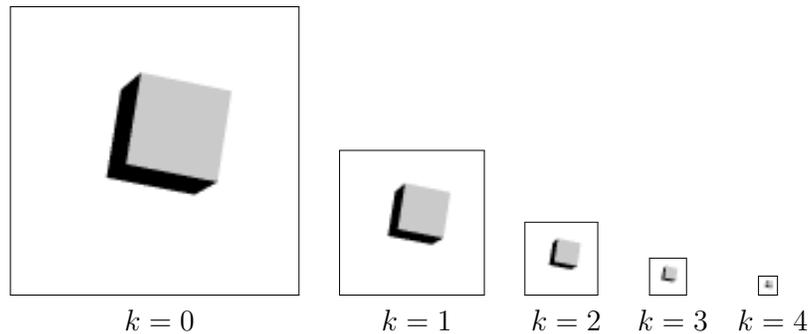


Figure 6.2: Example of a geometric description  $G$  rendered at different scales with  $\mu = 1/2$ .

### 6.2.2 Anti-aliasing

Tracing a single ray per pixel usually generates abrupt transitions along the edges of objects (see figure 6.3). This effect is known in computer graphics as *aliasing* or *jaggies*. As we observed in section 4.5, these discontinuities in the object's outline and pixel values

cause discontinuities in the discrepancy function, which are problematic for the nonlinear optimization procedure (Appendix A). Therefore, the RENDER procedure uses super-sampling in order to reduce those artifacts. Specifically, RENDER traces  $n_r^2$  rays in each pixel of  $B^{(k)}$ , through a regular  $n_r \times n_r$  array of sample points inside the pixel, and sets the pixel's value to the average of the colors obtained with those rays.

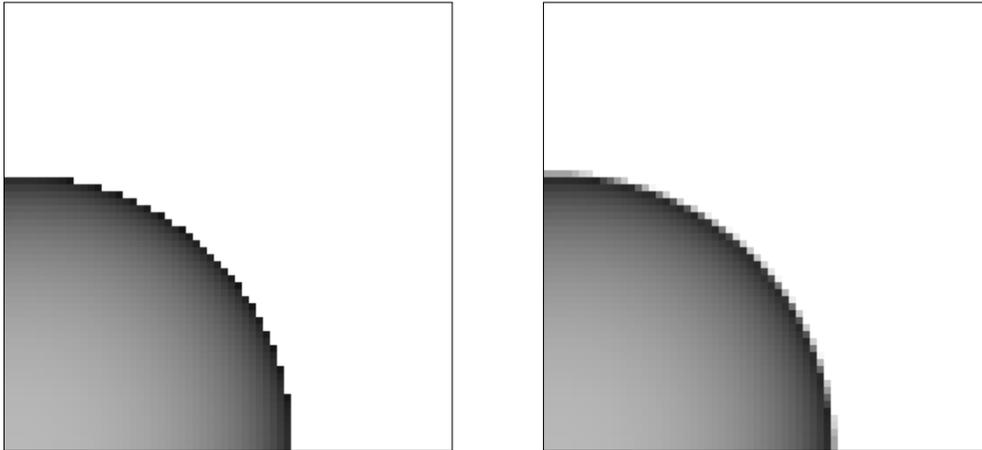


Figure 6.3: Image of a SPHERE primitive rendered with  $1 \times 1$  rays per pixel (left) and image rendered with  $10 \times 10$  rays per pixel (right).

# Chapter 7

## Test models

In this chapter we describe the organism models used in our tests. In all tests we assumed that models have their centers on the plane  $Z = 0$ , since it is impossible to determine the precise depth coordinate  $Z$  of the organism in the optical microscope images.

### 7.1 The ball model

The simplest model used in our tests is a perfectly spherical “organism” called **ball**. It has only three parameters: the position  $(X, Y)$  and the radius  $R$ . Its geometric description is a single instance of the SPHERE primitive shape, with translation vector  $t = (X, Y, 0)$  and diagonal matrix  $M = RI$ , where  $I$  is the  $3 \times 3$  identity matrix. See figure 7.1.

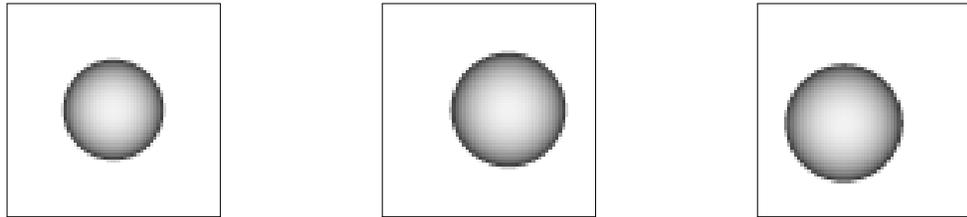


Figure 7.1: Three rendered images of the **ball** model.

### 7.2 The box model

The **box** model is an organism with cubical shape in general orientation. . It has six parameters: the position of the center  $(X, Y)$ , the radius (half-diagonal)  $S$  of the diagonal, and the rotation angles  $\theta_X, \theta_Y$  and  $\theta_Z$  that are applied successively about  $X, Y$  and  $Z$

axes. Each angle is allowed vary in the range  $[0, 90^\circ]$  (although two or more combinations of angles may be result in the same apparent pose). The geometric description of the `box` model is a single instance of the `CUBE` primitive (which is centered at  $(0, 0, 0)$  and unit diagonal), with position vector  $t = (X, Y, 0)$  and the following transformation matrix  $M$

$$M = \begin{pmatrix} \cos \theta_Z & -\sin \theta_Z & 0 \\ \sin \theta_Z & \cos \theta_Z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta_Y & 0 & \sin \theta_Y \\ 0 & 1 & 0 \\ -\sin \theta_Y & 0 & \cos \theta_Y \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_X & -\sin \theta_X \\ 0 & \sin \theta_X & \cos \theta_X \end{pmatrix} \begin{pmatrix} S & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & S \end{pmatrix}$$

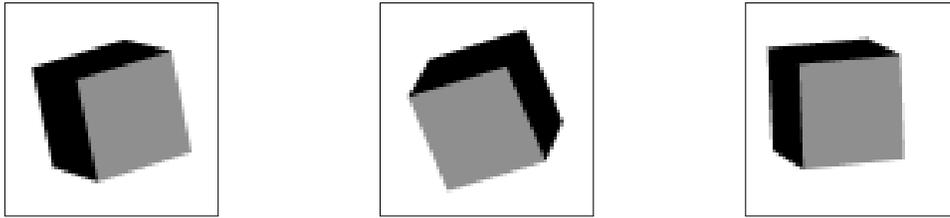
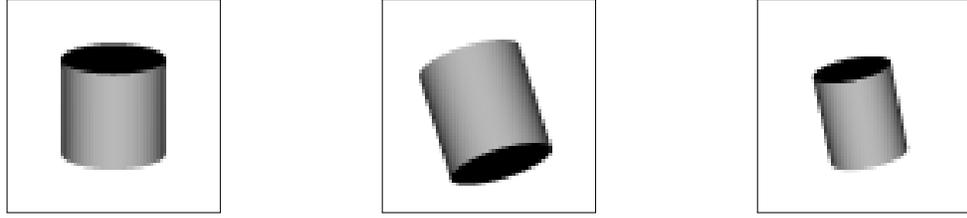


Figure 7.2: Three rendered images of the box model.

### 7.3 The `cyl` model

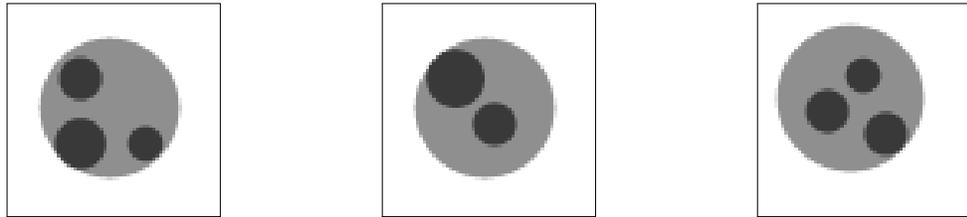
The `cyl` model is a perfectly cylindrical organism in general orientation. It is a reasonably acceptable model for the *Paralia sulcata diatom sp.* It has six parameters: the position of the center  $(X, Y)$ , the height  $H$ , the radius  $R$  and the rotations angles  $\theta_X$  (in the range  $[-90^\circ, 90^\circ]$ ) and  $\theta_Z$  (in the range  $[0, 180^\circ]$ ) that are applied successively about  $X$  and  $Z$  axes. The geometry of the `cyl` model is a single instance of the `CYLINDER` primitive, with position vector  $t = (X, Y, 0)$  and the following transformation matrix  $M$

$$M = \begin{pmatrix} \cos \theta_Z & -\sin \theta_Z & 0 \\ \sin \theta_Z & \cos \theta_Z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_X & -\sin \theta_X \\ 0 & \sin \theta_X & \cos \theta_X \end{pmatrix} \begin{pmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & H \end{pmatrix}$$

Figure 7.3: Three rendered images of the `cyl` model.

## 7.4 The `volv` model

The `volv` model is a simple model for the *Volvox* sp. colonial flagellate [45]. It consists of a hollow translucent sphere containing a variable number (zero or more) of smaller spheres. It has one discrete meta-parameter  $q$  that is the number of internal spheres. For a fixed  $q$ , the `volv` model has  $n = 3(q + 1)$  parameters, which are the radius  $R_0$  and position  $(X_0, Y_0)$  of the main sphere, and the radii  $R_i$  and the positions  $(X_i, Y_i)$  of internal spheres, for  $i = 1, 2, 3, \dots, q$ . The geometric description of a `volv` model is the union of  $q + 1$  instances of the `SPHERE` primitive with translation vectors  $t_i = (X_i, Y_i, 0)$  and matrices  $M_i = R_i I$ , for  $i = 0, \dots, n$ . See figure 7.4.

Figure 7.4: Three rendered images of the `volv` model.

## 7.5 The `prmc` model

The `prmc` model is meant to describe the protozoon *Paramecium* sp. [14]. The `prmc` model has  $n = 8$  parameters: the center of its body  $(X, Y)$ , four width parameters  $l_1, \dots, l_4$ , the total length  $h$ , and the rotation angle  $\theta_Z$  about the axis  $Z$ . The `prmc` model is not bilaterally symmetric.

The geometric description of a `prmc` model, as returned by the `BUILDGEOMETRY` method consists of 4 flat bi-cubic `BEZIER` patches  $b^{[ij]}$  that compose the body where  $i, j \in \{0, 1\}$ , contained on the plane  $Z = 0$ . See figure 7.5.

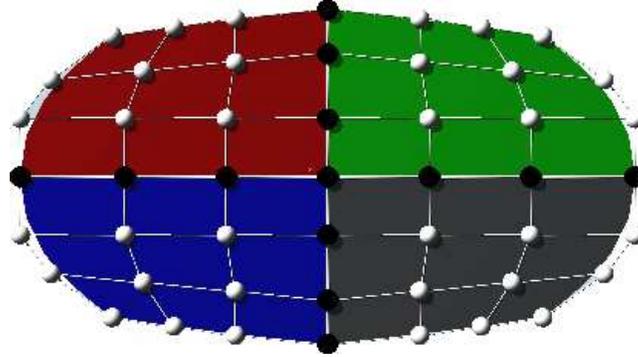


Figure 7.5: Bézier patches and control points.

The parameters  $l_1, \dots, l_4$  and the total length  $h$  are mapped to a certain subset of the Bézier control points of those patches, as shown in figure 7.5. The remaining Bézier control points are computed from these so as to maintain  $C_0$  and  $C_1$  parametric continuity between the patches.

Specifically let  $b_{rs}^{[ij]}$  be the Bézier control point  $r, s$  ( for  $0 \leq r, s \leq 3$ ) of patch  $b^{[ij]}$  (for  $0 \leq i, j \leq 1$ ). The method sets certain *primary control points* (shown in white in figure 7.5) as specified in table 7.1.

These *primary points* are shown in white in figure 7.5. The remaining points (black) are set to the average of neighboring primary points, so that the surface is  $C_1$  (parametric) everywhere. Some images of the `prmc` model can be seen in figure 7.6.

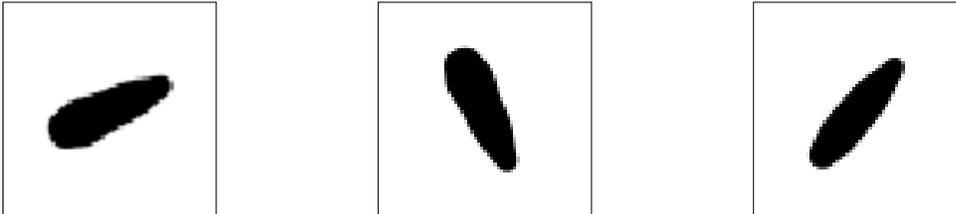


Figure 7.6: Three rendered images of the `prmc` model.

$$\begin{array}{lll}
b_{00}^{[00]} = \begin{pmatrix} -h/2, 0, 0 \end{pmatrix} & b_{01}^{[00]} = \begin{pmatrix} -h/3, 0, 0 \end{pmatrix} & b_{02}^{[00]} = \begin{pmatrix} -h/6, 0, 0 \end{pmatrix} \\
b_{10}^{[00]} = \begin{pmatrix} -h/2, l_1/3, 0 \end{pmatrix} & b_{11}^{[00]} = \begin{pmatrix} -h/3, l_1/3, 0 \end{pmatrix} & b_{12}^{[00]} = \begin{pmatrix} -h/6, l_1/3, 0 \end{pmatrix} \\
b_{20}^{[00]} = \begin{pmatrix} -h, l_1/3, 0 \end{pmatrix} & b_{21}^{[00]} = \begin{pmatrix} -h/3, l_1/3, 0 \end{pmatrix} & b_{22}^{[00]} = \begin{pmatrix} -h/6, l_1/3, 0 \end{pmatrix} \\
\\
b_{03}^{[01]} = \begin{pmatrix} h/2, 0, 0 \end{pmatrix} & b_{02}^{[01]} = \begin{pmatrix} h/3, 0, 0 \end{pmatrix} & b_{01}^{[01]} = \begin{pmatrix} h/6, 0, 0 \end{pmatrix} \\
b_{13}^{[01]} = \begin{pmatrix} h/2, l_2/3, 0 \end{pmatrix} & b_{12}^{[01]} = \begin{pmatrix} h/3, l_2/3, 0 \end{pmatrix} & b_{11}^{[01]} = \begin{pmatrix} h/6, l_2/3, 0 \end{pmatrix} \\
b_{23}^{[01]} = \begin{pmatrix} h, l_2/3, 0 \end{pmatrix} & b_{22}^{[01]} = \begin{pmatrix} h/3, l_2/3, 0 \end{pmatrix} & b_{21}^{[01]} = \begin{pmatrix} h/6, l_2/3, 0 \end{pmatrix} \\
\\
b_{00}^{[10]} = \begin{pmatrix} -h/2, 0, 0 \end{pmatrix} & b_{01}^{[10]} = \begin{pmatrix} -h/3, 0, 0 \end{pmatrix} & b_{02}^{[10]} = \begin{pmatrix} -h/6, 0, 0 \end{pmatrix} \\
b_{10}^{[10]} = \begin{pmatrix} -h/2, -l_3/3, 0 \end{pmatrix} & b_{11}^{[10]} = \begin{pmatrix} -h/3, -l_3/3, 0 \end{pmatrix} & b_{12}^{[10]} = \begin{pmatrix} -h/6, -l_3/3, 0 \end{pmatrix} \\
b_{20}^{[10]} = \begin{pmatrix} -h, -l_3/3, 0 \end{pmatrix} & b_{21}^{[10]} = \begin{pmatrix} -h/3, -l_3/3, 0 \end{pmatrix} & b_{22}^{[10]} = \begin{pmatrix} -h/6, -l_3/3, 0 \end{pmatrix} \\
\\
b_{03}^{[11]} = \begin{pmatrix} h/2, 0, 0 \end{pmatrix} & b_{02}^{[11]} = \begin{pmatrix} h/3, 0, 0 \end{pmatrix} & b_{01}^{[11]} = \begin{pmatrix} h/6, 0, 0 \end{pmatrix} \\
b_{13}^{[11]} = \begin{pmatrix} h/2, -l_4/3, 0 \end{pmatrix} & b_{12}^{[11]} = \begin{pmatrix} h/3, -l_4/3, 0 \end{pmatrix} & b_{11}^{[11]} = \begin{pmatrix} h/6, -l_4/3, 0 \end{pmatrix} \\
b_{23}^{[11]} = \begin{pmatrix} h, -l_4/3, 0 \end{pmatrix} & b_{22}^{[11]} = \begin{pmatrix} h/3, -l_4/3, 0 \end{pmatrix} & b_{21}^{[11]} = \begin{pmatrix} h/6, -l_4/3, 0 \end{pmatrix}
\end{array}$$

Table 7.1: Primary control points of the *prmc*.

## 7.6 The *cycl* model

The *cycl* model is meant to describe a female specimen of the crustacean *Cyclops* sp. [38] with egg sacs. See figure 7.7. The *cycl* model has  $n = 11$  parameters: the center of its body  $(X, Y)$ , two segment width parameters  $l_1$  and  $l_2$ , the total length  $h$ , the rotation angle  $\theta_Z$  about the axis  $Z$ , the relative longitudinal position  $t$  of the eggs sacs, the length and width  $e_x$  and  $e_y$  of the eggs sacs, and their inclination angles  $a_1$  and  $a_2$  relative to the body axis.

The geometric description of a *cycl* model, as returned by the BUILDGEOMETRY method consists of 4 flat bi-cubic BEZIER patches that represent the body, contained on the plane  $Z = 0$ ; and two ellipsoids that represent the egg sacs. The parameters  $l_1$  and  $l_2$  and the total length  $h$  are mapped to a certain subset of the Bézier control points of those patches. Unlike the *prmc* model, the *cycl* model has bilateral symmetry about the  $X$ -axis. The remaining Bézier control points are computed from these. Specifically let  $b_{rs}^{[ij]}$  be the Bézier control point  $r, s$  (for  $0 \leq r, s \leq 3$ ) of patch  $b^{[ij]}$  (for  $0 \leq i, j \leq 1$ ). The method sets certain *primary control points* (shown in white in figure 7.7) as specified in table 7.2. Some images of the *cycl* can be seen in figure 7.8. In optical microscope the *cyclops* tend to lie with the eggs sacs parallel to  $XY$  plane, so that we felt unnecessary to allow for rotation about the  $X$ -axis.

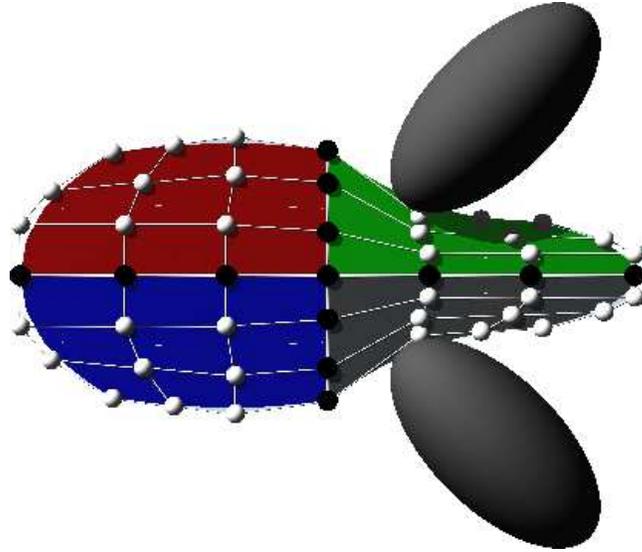


Figure 7.7: Bézier patches and control points.

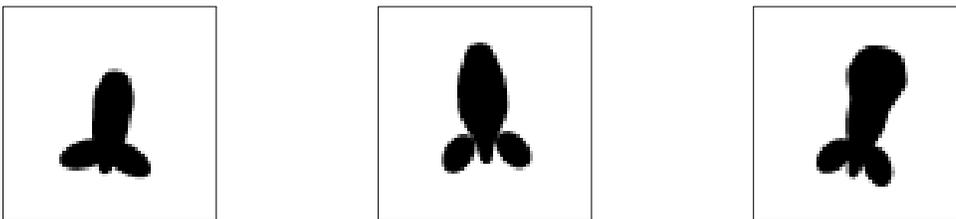


Figure 7.8: Three rendered images of the `cycl` model.

$$\begin{array}{lll}
b_{00}^{[00]} = \begin{pmatrix} -h/2, 0, 0 \end{pmatrix} & b_{01}^{[00]} = \begin{pmatrix} -h/3, 0, 0 \end{pmatrix} & b_{02}^{[00]} = \begin{pmatrix} -h/6, 0, 0 \end{pmatrix} \\
b_{10}^{[00]} = \begin{pmatrix} -h/2, l_1/3, 0 \end{pmatrix} & b_{11}^{[00]} = \begin{pmatrix} -h/3, l_1/3, 0 \end{pmatrix} & b_{12}^{[00]} = \begin{pmatrix} -h/6, l_1/3, 0 \end{pmatrix} \\
b_{20}^{[00]} = \begin{pmatrix} -h, l_1/3, 0 \end{pmatrix} & b_{21}^{[00]} = \begin{pmatrix} -h/3, l_1/3, 0 \end{pmatrix} & b_{22}^{[00]} = \begin{pmatrix} -h/6, l_1/3, 0 \end{pmatrix} \\
\\
b_{03}^{[01]} = \begin{pmatrix} h/2, 0, 0 \end{pmatrix} & b_{02}^{[01]} = \begin{pmatrix} h/3, 0, 0 \end{pmatrix} & b_{01}^{[01]} = \begin{pmatrix} h/6, 0, 0 \end{pmatrix} \\
b_{13}^{[01]} = \begin{pmatrix} h/2, l_2/3, 0 \end{pmatrix} & b_{12}^{[01]} = \begin{pmatrix} h/3, l_2/3, 0 \end{pmatrix} & b_{11}^{[01]} = \begin{pmatrix} h/6, l_2/3, 0 \end{pmatrix} \\
b_{23}^{[01]} = \begin{pmatrix} h, l_2/3, 0 \end{pmatrix} & b_{22}^{[01]} = \begin{pmatrix} h/3, l_2/3, 0 \end{pmatrix} & b_{21}^{[01]} = \begin{pmatrix} h/6, l_2/3, 0 \end{pmatrix} \\
\\
b_{00}^{[10]} = \begin{pmatrix} -h/2, 0, 0 \end{pmatrix} & b_{01}^{[10]} = \begin{pmatrix} -h/3, 0, 0 \end{pmatrix} & b_{02}^{[10]} = \begin{pmatrix} -h/6, 0, 0 \end{pmatrix} \\
b_{10}^{[10]} = \begin{pmatrix} -h/2, -l_3/3, 0 \end{pmatrix} & b_{11}^{[10]} = \begin{pmatrix} -h/3, -l_3/3, 0 \end{pmatrix} & b_{12}^{[10]} = \begin{pmatrix} -h/6, -l_3/3, 0 \end{pmatrix} \\
b_{20}^{[10]} = \begin{pmatrix} -h, -l_3/3, 0 \end{pmatrix} & b_{21}^{[10]} = \begin{pmatrix} -h/3, -l_3/3, 0 \end{pmatrix} & b_{22}^{[10]} = \begin{pmatrix} -h/6, -l_3/3, 0 \end{pmatrix} \\
\\
b_{03}^{[11]} = \begin{pmatrix} h/2, 0, 0 \end{pmatrix} & b_{02}^{[11]} = \begin{pmatrix} h/3, 0, 0 \end{pmatrix} & b_{01}^{[11]} = \begin{pmatrix} h/6, 0, 0 \end{pmatrix} \\
b_{13}^{[11]} = \begin{pmatrix} h/2, -l_4/3, 0 \end{pmatrix} & b_{12}^{[11]} = \begin{pmatrix} h/3, -l_4/3, 0 \end{pmatrix} & b_{11}^{[11]} = \begin{pmatrix} h/6, -l_4/3, 0 \end{pmatrix} \\
b_{23}^{[11]} = \begin{pmatrix} h, -l_4/3, 0 \end{pmatrix} & b_{22}^{[11]} = \begin{pmatrix} h/3, -l_4/3, 0 \end{pmatrix} & b_{21}^{[11]} = \begin{pmatrix} h/6, -l_4/3, 0 \end{pmatrix}
\end{array}$$

Table 7.2: Primary control points of the *cycl*.

## 7.7 The `eleg` model

The `eleg` model is meant to describe the worm *Caenorhabditis Elegans* sp. [54]. See figure 7.9. It has a meta-parameter  $q \geq 2$  that determines the number of segments in the model which in turn determines the number of ways in which it can bend. For a fixed  $q$ , the `eleg` model has  $n = 2q + 5$  parameters, which are the center  $(X, Y)$  of the body, its length  $l$ , its diameter  $d$ , a rotation angle  $\theta_Z$  about the axis  $Z$ , and  $2q$  bending parameters  $\alpha_1, \dots, \alpha_q$  and  $\beta_1, \dots, \beta_q$ .

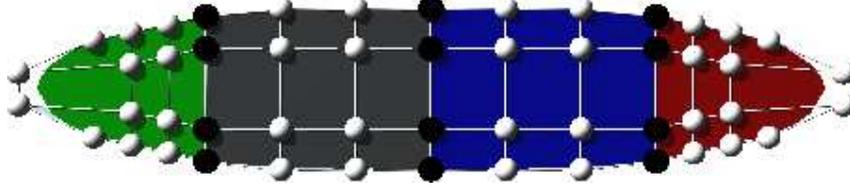


Figure 7.9: Bézier patches and control points for the `eleg` model with  $q = 3$  (4 patches).

The `BUILDGEOMETRY` procedure assumes that the worm has cylindrical symmetry and can bend only parallel to  $X, Y$  plane. Therefore the geometric description consists of  $q + 1$  flat bi-cubic BEZIER patches. See figure 7.9.

Some of the control points are computed from the bending parameters to achieve the desired shape; these are the *primary points* shown in white in figure 7.9. First the procedure defines an *axial curve*  $(u, \Omega(u))$  for  $u \in [-1, +1]$  where

$$\Omega(u) = \sum_{k=1}^q [\alpha_k \cos(u k \frac{\pi}{2}) + \beta_k \sin(u k \frac{\pi}{2})] \quad (7.1)$$

At each  $u$ , the normal vector  $\vec{n}(u)$  is  $(\Omega'(u), -1) / \sqrt{1 + \Omega'(u)^2}$  where  $\Omega'(u)$  is the derivative of  $\Omega(u)$ , namely

$$\Omega'(u) = \sum_{k=1}^q [-\alpha_k k \frac{\pi}{2} \sin(u k \frac{\pi}{2}) + \beta_k k \frac{\pi}{2} \cos(u k \frac{\pi}{2})] \quad (7.2)$$

Then the procedure maps each primary point  $(u, v)$  to point  $(u', v')$

$$(u', v') = \frac{l}{2}(u, \Omega(u)) + \frac{vd}{2}\vec{n}(u) \quad (7.3)$$

The remaining Bézier points are computed from those primary points so that the surface is  $C_1$  (parametric) at the joints.

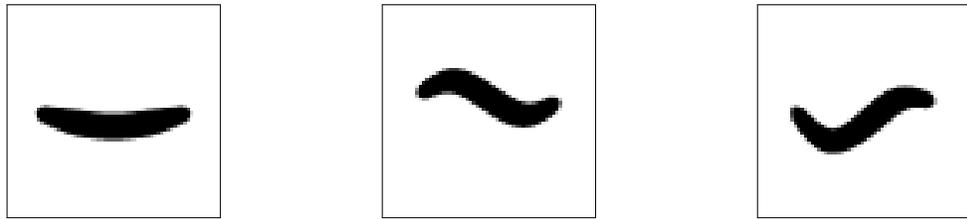


Figure 7.10: Three rendered images of the `eleg` model for  $q = 3$  (4 patches).



# Chapter 8

## Experimental evaluation

In this chapter we describe various experimental tests of the MSFIT algorithm.

### 8.1 Test datasets

In these tests we used two sets of input data, **SINT** and **REAL** both with  $N_M = 7$  models (described in Chapter 7) and  $N_G = 10$  input images for each model.

The set **SINT** consists of synthetic input images with white background and size  $n_x \times n_y = 32 \times 32$  pixels, generated by our ray tracer, with a randomly chosen parameter vector. The set **REAL** consists of actual digital photos or microscope images of size  $n_x \times n_y = 32 \times 32$  obtained from various Internet sites, manually scaled and cropped to uniform size. See figures 8.1 and 8.2

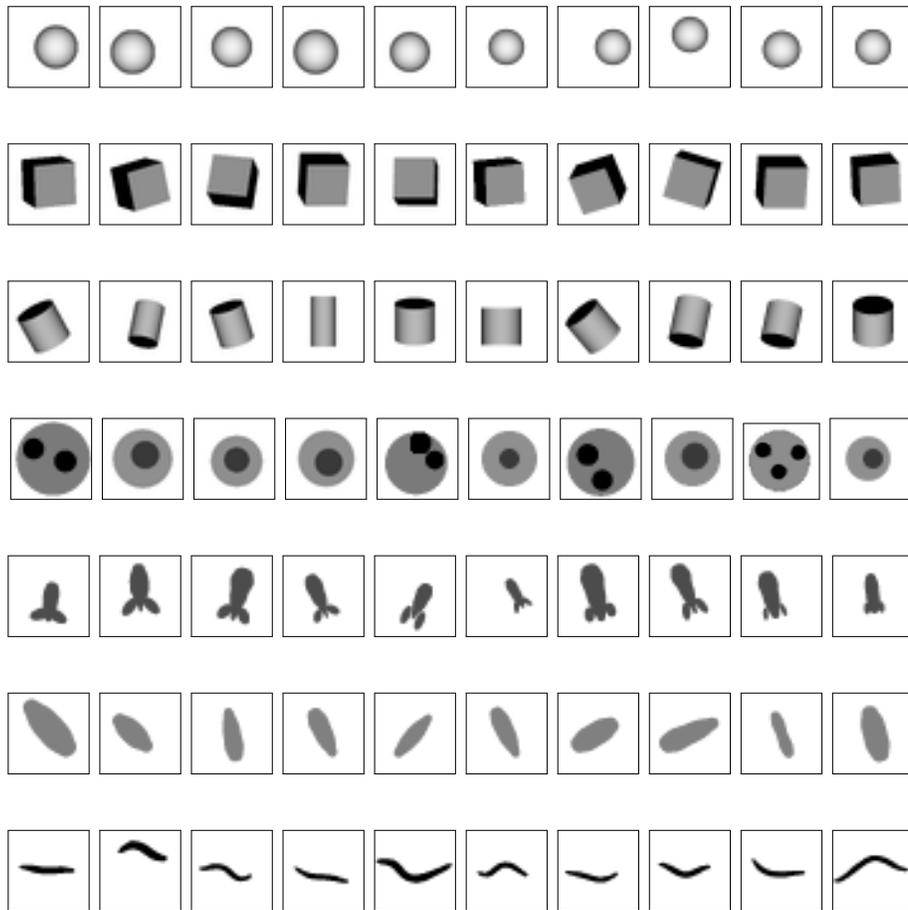


Figure 8.1: Images of the SINT dataset generated from models ball, box, cyl, volv, cycl, prmc and eleg (top to bottom).

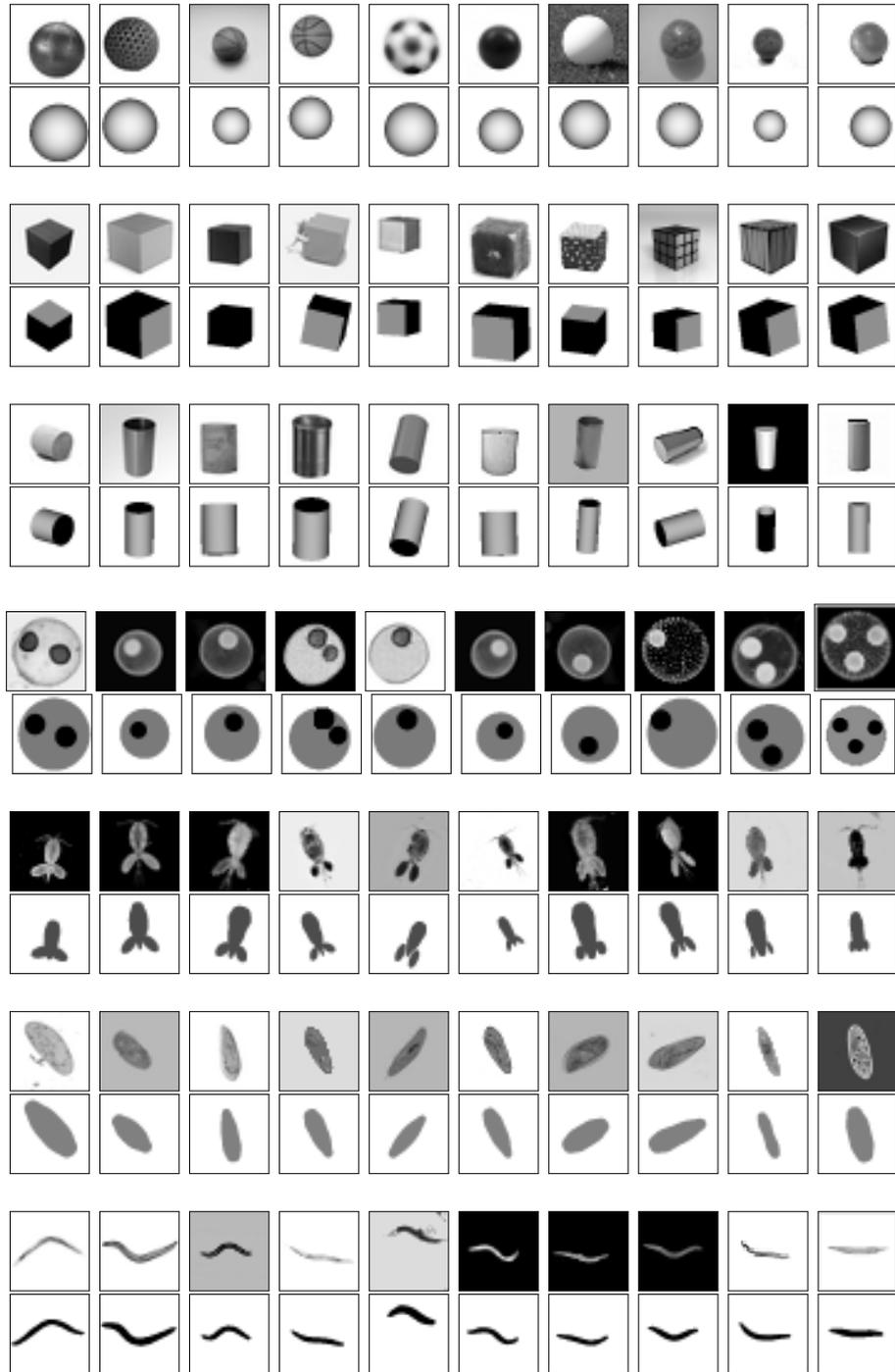


Figure 8.2: Images of REAL dataset with real images of “organisms” fitting the ball, box, cyl, volv, cycl, prmc and eleg models (top to bottom). For each model, the first row shows the digital photos of the actual “organism”, and the second row shows the synthetic images generated with the corresponding template parameter vectors  $p^*$ .

## 8.2 Automatic tests evaluation

In order to evaluate the performance of our algorithm on a significant number of test images we had to develop a method to automatically check whether the results were correct or not. For this purpose, we defined two metrics  $d_{\text{avg}}$  and  $d_{\text{max}}$  to measure the distance between each vector of parameters  $p$  obtained by the algorithm and the *template*  $p^*$ , the correct set of parameters for each input image. These metrics take into account the magnitude of the effect of each parameter on the model's image. Specifically, for each scale  $k$ , we define

$$d_{\text{avg}}^{(k)}(p, p^*) = \sqrt{\frac{1}{n} \sum_i^n \frac{(p^*[i] - p[i])^2}{[(\delta_*^{(k)}[i])^2 + (\delta^{(k)}[i])^2]/2}} \quad (8.1)$$

$$d_{\text{max}}^{(k)}(p, p^*) = \max_{0 \leq i \leq n-1} \frac{|p^*[i] - p[i]|}{\sqrt{[(\delta_*^{(k)}[i])^2 + (\delta^{(k)}[i])^2]/2}} \quad (8.2)$$

In both formulas,  $\delta^{(k)}[i]$  is a value such that the change of parameter  $p[i]$  to  $p[i] + \delta^{(k)}[i]$  causes some part of the model's projection to move by about 1/2 pixel in the scale  $k$ . Therefore, the values of  $d_{\text{avg}}^{(k)}$  and  $d_{\text{max}}^{(k)}$  can be interpreted as a distance in pixels at the scale  $k$  between the projections of  $p$  and  $p^*$ . In our tests we asked MSFIT to return a single parameter vector ( $t = 1$ ) and we considered that a run of the algorithm was successful (was a *hit*) if  $d_{\text{avg}}^{(0)} \leq 1.25$  and  $d_{\text{max}}^{(0)} \leq 1.75$ .

The automatic comparison of the parameters must also take into account that some models have symmetries, so that there are several vectors  $p$  quite distinct from each other that produce the same image. In these cases, our evaluation procedure generates the various parameter vectors equivalent to the template  $p^*$  and considers the run a success if at least one of those vector  $p$  satisfies the above criteria.

For the set **SINT**, the template  $p^*$  of each test image is the vector of parameters used in their synthesis. For the set **REAL**, the vector of parameters (template) of each image was obtained manually from a vector of initial parameter obtained by MSFIT. This vector was manually adjusted until the corresponding synthetic image visually is as close as possible to the actual image.

## 8.3 Average performance

Since the algorithm is not deterministic, we executed the algorithm  $N_E = 10$  times with different seeds for the random number generator, on each of  $N_M 2 N_G$  test images. We then computed the *hit rate*  $H_A$  for each specific model and each input image type (**REAL** or **SINT**) as the number of hits divided by the corresponding number of runs,  $N_G \times N_E$ .

For each model and input image type we also computed the standard deviation  $H_D$  of the hit rate over the  $N_G$  images, averaged over the  $N_E$  runs of that image. As a measure of the cost of the algorithm, we use the average number of rays cast  $N_R$  by all calls of the `RENDER` procedure.

## 8.4 Internal parameters

The `MSFIT` algorithm has several internal parameters that can be tuned to improve its hit rate and/or running time. The main are:

- $L_k$ , the maximum size of the queue  $Q$  at each level  $k \geq 1$ ;
- $n_r$ , an integer that defines the number of rays ( $n_r \times n_r$ ) cast per pixel;
- $\mu$ , image scaling factor of the pyramid;
- $D$ , the formula used for calculate the discrepancy  $c.d$  of each candidate  $c$  (either  $D_S$  or  $D_H$ , see section 5.2.1)
- $n_i$ , the number of iterations of our nonlinear method.

The values of these parameters were chosen after many experiments on the datasets described in section 8.2 so as to maximize the hit rate without excessive computational cost. The chosen values were:  $L_k = 1.0$ ,  $\mu = 1/\sqrt{2}$ ,  $n_r = 4$ ,  $n_i = 4$ , and the use of  $D_S$  to compute the discrepancy of the candidates.

## 8.5 Results

In this section we present some tests that show the influence of the chosen values of the internal parameters. Namely we vary the value of each parameter around the chosen value, while fixing all other internal parameters at their chosen values. Needless to say, these experiments do not prove that the chosen values are optimal, since the performance may still be improved by changing two or more them at the same time.

### 8.5.1 Effect of the distance estimator $D$

In this subsection we show that  $D_S$  is slightly better than  $D_H$ , given the other parameter values chosen. For that we must find the optimal value of the internal parameter  $\beta$  of that defines the weight  $\lambda_k = \beta^{-k}$  of the  $D_H$  in the formula (5.3). For this test we vary  $\beta$  between 0.25 and 2.5 with  $D = D_H$ . See tables 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8 and figures 8.3 and 8.4.

$\beta = 0.25$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	26.35	20%	4%	25.65	36%	6%
box	23.78	86%	6%	22.88	92%	5%
cycl	92.15	44%	11%	89.55	62%	11%
eleg	68.16	26%	7%	69.48	56%	6%
prmc	31.06	62%	9%	29.25	78%	8%
ball	6.66	82%	5%	6.99	96%	3%
volv	22.86	68%	5%	23.99	96%	6%
<b>Total</b>	38.72	55%	7%	38.26	74%	6 %

Table 8.1: Results obtained by MSFIT with  $D = D_H$  and  $\beta = 0.25$ .

$\beta = 0.5$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	26.51	22%	5%	25.57	44%	7%
box	23.64	86%	3%	23.66	84%	7%
cycl	88.27	58%	8%	87.27	62%	8%
eleg	69.58	32%	5%	68.54	44%	7%
prmc	31.07	80%	9%	29.82	74%	11%
ball	6.75	80%	4%	7.11	96%	3%
volv	23.69	68%	9%	23.71	98%	3%
<b>Total</b>	38.50	61%	6%	37.95	72%	7 %

Table 8.2: Results obtained by MSFIT with  $D = D_H$  and  $\beta = 0.5$ .

$\beta = 0.75$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	25.75	20%	6%	24.66	58%	8%
box	22.81	76%	3%	23.08	92%	5%
cycl	83.78	64%	3%	88.51	82%	13%
eleg	69.84	42%	7%	72.51	52%	10%
prmc	30.11	68%	7%	30.50	86%	11%
ball	6.67	86%	8%	6.96	98%	3%
volv	22.24	70%	6%	24.17	98%	3%
<b>Total</b>	37.31	61%	6%	38.63	81%	8 %

Table 8.3: Results obtained by MSFIT with  $D = D_H$  and  $\beta = 0.75$ .

$\beta = 1.0$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	24.37	24%	11%	24.13	44%	10%
box	23.61	92%	5%	24.20	90%	6%
cycl	82.84	60%	6%	90.89	72%	3%
eleg	68.74	42%	5%	68.80	46%	8%
prmc	31.55	80%	9%	31.33	86%	7%
ball	6.64	90%	4%	7.15	96%	3%
volv	22.18	64%	3%	24.74	100%	0%
<b>Total</b>	37.13	65%	6%	38.75	76%	5 %

Table 8.4: Results obtained by MSFIT with  $D = D_H$  and  $\beta = 1.0$ .

$\beta = 1.5$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	24.45	24%	7%	24.69	62%	5%
box	21.68	84%	7%	22.55	88%	7%
cycl	81.96	68%	5%	84.63	82%	9%
eleg	67.42	30%	6%	70.98	62%	8%
prmc	29.94	84%	7%	29.01	84%	6%
ball	6.78	86%	7%	7.26	98%	3%
volv	22.35	70%	4%	23.91	100%	0%
<b>Total</b>	36.37	64%	6%	37.58	82%	5 %

Table 8.5: Results obtained by MSFIT with  $D = D_H$  and  $\beta = 1.5$ .

$\beta = 2.0$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	24.74	36%	7%	24.07	58%	12%
box	23.25	86%	7%	23.54	92%	5%
cycl	80.66	66%	7%	79.26	80%	6%
eleg	69.00	38%	12%	70.88	52%	10%
prmc	30.60	84%	7%	30.62	96%	3%
ball	6.73	88%	7%	8.62	96%	3%
volv	22.95	64%	6%	23.67	100%	0%
<b>Total</b>	<b>36.85</b>	<b>66%</b>	<b>8%</b>	<b>37.24</b>	<b>82%</b>	<b>6 %</b>

Table 8.6: Results obtained by MSFIT with  $D = D_H$  and  $\beta = 2.0$ .

$\beta = 2.5$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	25.08	28%	14%	24.00	68%	12%
box	22.46	86%	3%	23.34	86%	3%
cycl	84.40	70%	4%	85.77	64%	7%
eleg	69.18	48%	7%	71.29	54%	13%
prmc	32.09	82%	7%	30.64	98%	3%
ball	6.59	86%	3%	7.27	98%	3%
volv	22.54	74%	3%	24.41	100%	0%
<b>Total</b>	<b>37.48</b>	<b>68%</b>	<b>6%</b>	<b>38.10</b>	<b>81%</b>	<b>6 %</b>

Table 8.7: Results obtained by MSFIT with  $D = D_H$  and  $\beta = 2.5$ .

$D = D_S$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	25.05	52%	5%	25.85	72%	7%
box	22.86	84%	10%	22.05	94%	3%
cycl	77.58	72%	5%	80.47	84%	10%
eleg	68.57	38%	5%	70.36	62%	8%
prmc	31.77	76%	10%	31.27	92%	8%
ball	6.36	80%	6%	7.01	96%	3%
volv	23.16	62%	5%	25.76	100%	0%
<b>Total</b>	<b>36.48</b>	<b>66%</b>	<b>7%</b>	<b>37.54</b>	<b>86%</b>	<b>6 %</b>

Table 8.8: Results obtained by MSFIT with  $D = D_S$ .

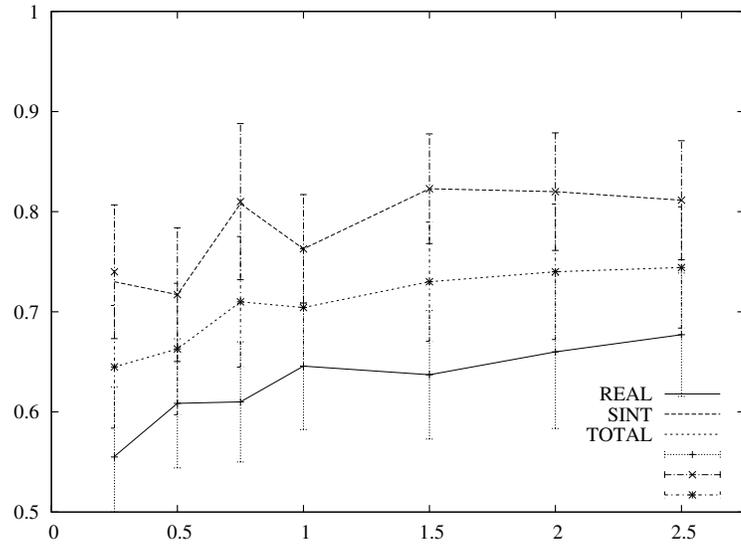


Figure 8.3: The effect of the  $\beta$  weight on the hit rate with  $D = D_H$ .

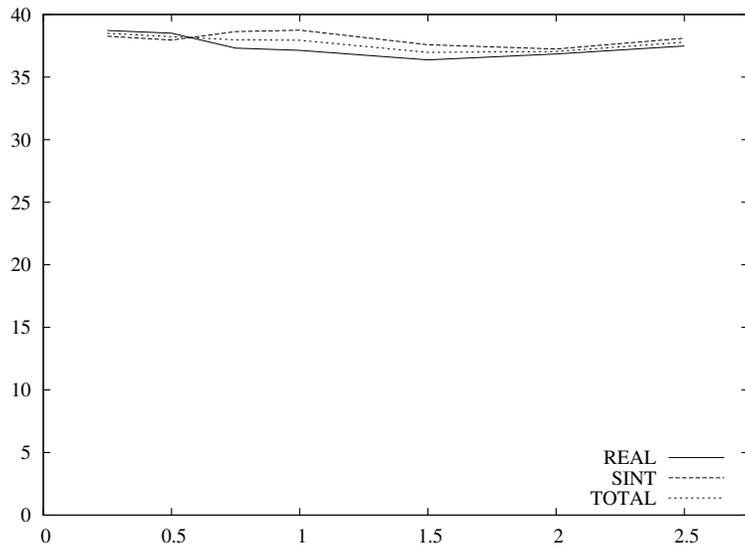


Figure 8.4: The effect of the  $\beta$  weight on the average number  $N_R$  of rays cast.

From these tables we conclude that  $\beta = 0.75$  provides the best performance when  $D = D_H$ . However a table 8.8 shows the performance with  $D = D_S$  is better.

### 8.5.2 Effect of image scale factor reduction

In this subsection we test the effect of the image scale factor reduction  $\mu$ . The usual choice  $\mu = 50\%$  is so far from the optimal, therefore we vary  $\mu$  between 60% and 80%. See tables 8.9, 8.10, 8.11, 8.12, 8.13, and figures 8.5 and 8.6. For the chosen settings of the others parameters, the best hit rate  $H_A$  is when  $\mu \approx 0.75$ , while the total cost  $N_R$  is surprisingly independent it of  $\mu$ . We chose  $\mu = \sqrt{1/2} = 0.7071$  so that even levels are scaled by exact powers of  $1/2$ .

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	24.08	18%	7%	23.82	44%	7%
box	22.26	58%	8%	20.62	84%	3%
cycl	96.82	46%	3%	96.93	52%	14%
eleg	70.01	26%	7%	68.89	54%	14%
prmc	30.73	70%	9%	31.17	94%	6%
ball	5.20	68%	5%	6.38	94%	3%
volv	19.54	60%	10%	19.53	94%	3%
<b>Total</b>	38.38	49%	7%	38.19	74%	7%

Table 8.9: Results obtained by MSFIT with  $\mu = 60\%$ .

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	22.89	22%	14%	22.90	52%	11%
box	21.35	70%	10%	21.22	92%	3%
cycl	78.60	56%	6%	82.93	80%	6%
eleg	65.99	40%	9%	68.33	52%	5%
prmc	29.33	80%	9%	30.39	94%	6%
ball	5.86	80%	4%	6.44	92%	3%
volv	21.07	64%	3%	23.32	100%	0%
<b>Total</b>	35.01	59%	8%	36.50	80%	5%

Table 8.10: Results obtained by MSFIT with  $\mu = 65\%$ .

$\mu = 70\%$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	24.63	26%	3%	25.22	62%	13%
box	22.36	78%	3%	21.69	96%	3%
cycl	81.59	72%	7%	83.96	80%	8%
eleg	69.50	40%	9%	69.88	46%	10%
prmc	31.58	90%	4%	29.76	90%	4%
ball	6.63	82%	5%	7.42	96%	3%
volv	23.01	60%	4%	24.91	100%	0%
<b>Total</b>	37.04	64%	5%	37.55	81%	6 %

Table 8.11: Results obtained by MSFIT with  $\mu = 70\%$ .

$\mu = 75\%$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	28.38	60%	4%	27.53	84%	3%
box	25.68	66%	3%	24.64	92%	5%
cycl	83.81	74%	6%	85.90	92%	5%
eleg	75.27	62%	10%	77.67	72%	8%
prmc	35.33	84%	6%	33.08	80%	4%
ball	7.95	80%	6%	7.91	92%	5%
volv	26.12	72%	5%	27.02	100%	0%
<b>Total</b>	40.36	71%	6%	40.53	87%	5 %

Table 8.12: Results obtained by MSFIT with  $\mu = 75\%$ .

$\mu = 80\%$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	32.50	66%	6%	33.69	88%	5%
box	28.12	80%	10%	27.25	86%	7%
cycl	90.45	74%	6%	95.23	90%	4%
eleg	86.24	56%	7%	88.24	46%	3%
prmc	41.04	84%	10%	39.97	80%	0%
ball	10.21	74%	7%	10.10	94%	3%
volv	31.00	82%	5%	31.34	100%	0%
<b>Total</b>	45.65	74%	7%	46.54	83%	3 %

Table 8.13: Results obtained by MSFIT with  $\mu = 80\%$ .

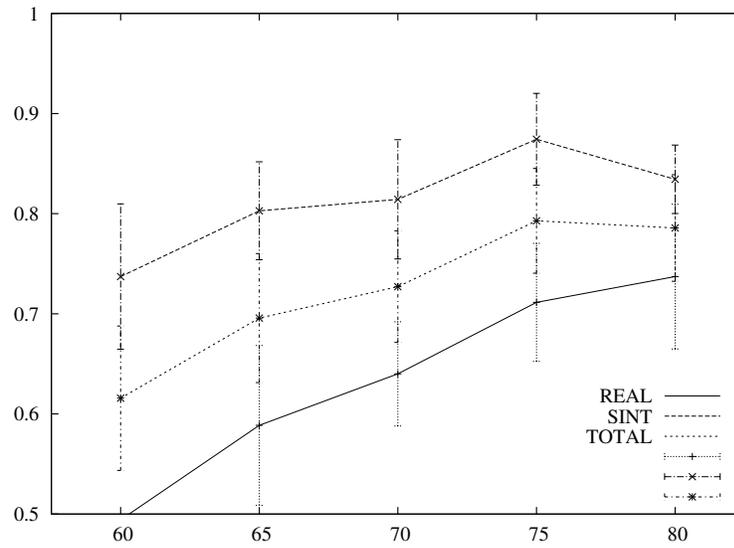


Figure 8.5: The effect of the scale factor  $\mu$  on the hit rate.

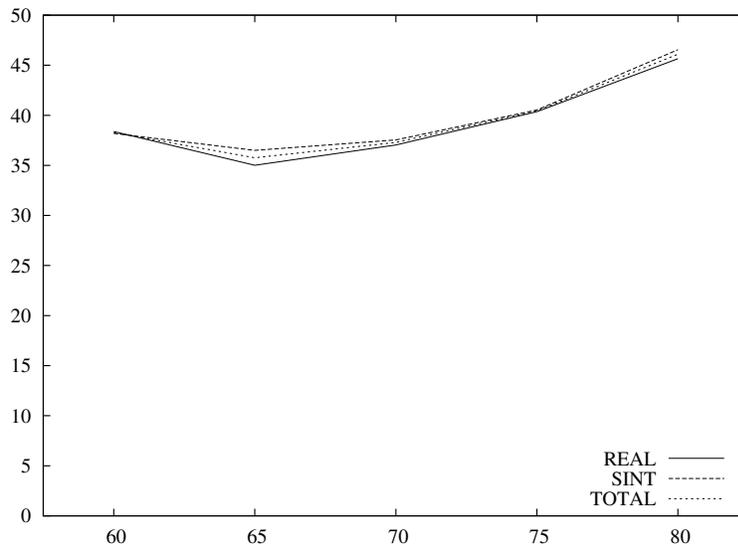


Figure 8.6: The effect of the scale factor  $\mu$  on the number of rays.

### 8.5.3 Effect of the number of rays cast per pixel

In this subsection we test the effect of the internal parameter  $n_r$  that defines the number of rays ( $n_r \times n_r$ ) cast per pixel. For this test we vary  $n_r$  between 2 and 6. See tables 8.14, 8.15, 8.16, 8.17, 8.18, and figures 8.7 and 8.8. As can be seen from these figures, the hit rate  $H_A$  appears increase slightly with the  $n_r$  (but less than its standard deviation  $H_D$ ) while the cost  $N_R$  appears to grow quadratically with  $n_r$ , as expected. Therefore we chose  $n_r = 4$ . However note that  $n_r = 3$  would provide nearly the same hit rate at half the cost.

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	6.36	46%	10%	6.31	66%	3%
box	5.57	70%	8%	5.31	90%	4%
cycl	19.54	66%	3%	19.52	72%	11%
eleg	17.87	38%	15%	17.42	68%	10%
prmc	7.34	76%	7%	7.47	72%	5%
ball	1.56	68%	5%	1.65	94%	3%
volv	5.65	66%	3%	6.06	100%	0%
<b>Total</b>	9.13	61%	7%	9.10	80%	5 %

Table 8.14: Results obtained by MSFIT with  $n_r = 2$ .

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	14.07	54%	10%	13.94	58%	5%
box	13.12	84%	6%	12.25	94%	6%
cycl	43.74	64%	11%	45.46	76%	7%
eleg	40.34	44%	6%	40.44	62%	7%
prmc	17.71	82%	8%	17.16	92%	3%
ball	3.60	74%	3%	3.81	98%	3%
volv	12.92	76%	3%	14.39	100%	0%
<b>Total</b>	20.78	68%	7%	21.06	83%	4 %

Table 8.15: Results obtained by MSFIT with  $n_r = 3$ .

$n_r = 4$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	25.05	52%	5%	25.85	72%	7%
box	22.86	84%	10%	22.05	94%	3%
cycl	77.58	72%	5%	80.47	84%	10%
eleg	68.57	38%	5%	70.36	62%	8%
prmc	31.77	76%	10%	31.27	92%	8%
ball	6.36	80%	6%	7.01	96%	3%
volv	23.16	62%	5%	25.76	100%	0%
<b>Total</b>	36.48	66%	7%	37.54	86%	6 %

Table 8.16: Results obtained by MSFIT with  $n_r = 4$ .

$n_r = 5$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	38.74	68%	7%	39.41	68%	8%
box	36.41	80%	9%	33.81	94%	3%
cycl	119.69	76%	3%	127.33	92%	5%
eleg	107.56	54%	7%	111.16	62%	7%
prmc	47.76	88%	5%	48.41	88%	5%
ball	10.03	80%	8%	10.48	98%	3%
volv	35.32	62%	10%	39.10	100%	0%
<b>Total</b>	56.50	73%	7%	58.53	86%	5 %

Table 8.17: Results obtained by MSFIT with  $n_r = 5$ .

$n_r = 6$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	53.97	62%	9%	56.54	68%	3%
box	51.33	76%	7%	49.52	96%	3%
cycl	175.54	78%	8%	182.93	88%	5%
eleg	154.32	50%	4%	156.55	64%	11%
prmc	70.03	86%	7%	69.10	94%	6%
ball	14.30	86%	6%	15.42	96%	3%
volv	52.20	72%	11%	54.17	98%	3%
<b>Total</b>	81.67	73%	8%	83.46	86%	5 %

Table 8.18: Results obtained by MSFIT with  $n_r = 6$ .

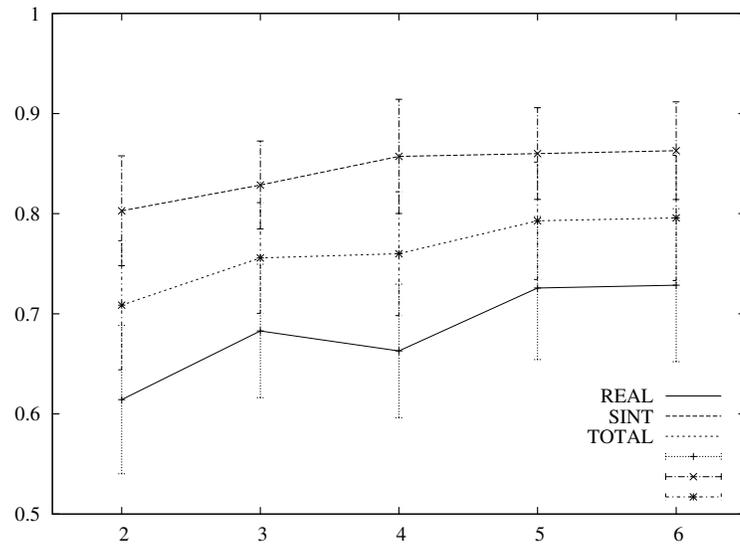


Figure 8.7: The effect of the parameter  $n_r$  on the hit rate.

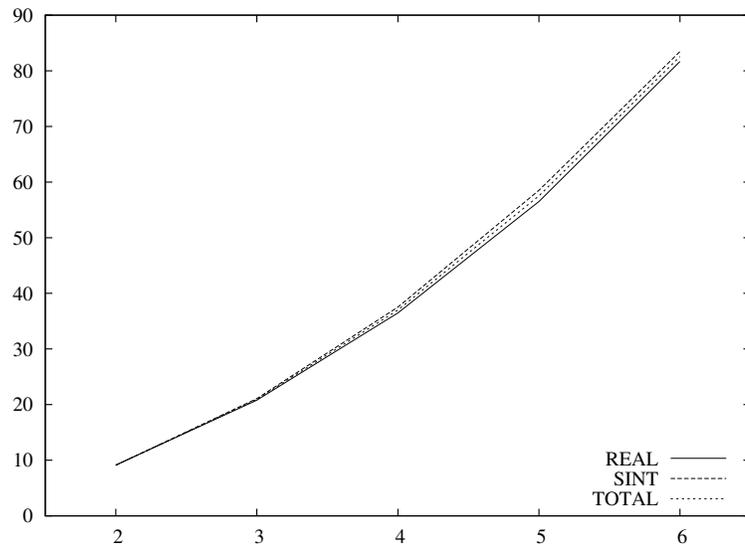


Figure 8.8: The effect of the parameter  $n_r$  on the computational cost.

### 8.5.4 Effect of the number of iterations

In this subsection we test the effect of the internal parameter  $n_i$  that defines the number of iterations executed by our nonlinear optimization method. For this test we vary  $n_i$  between 2 and 8. See tables 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25 and figures 8.9 and 8.10. As the plots show, the hit rate appears to be maximum for  $n_i$  around 6. However the cost grows linearly with  $n_i$ . Therefore we chose  $n_i = 5$  as a good compromise between accuracy and efficiency.

$n_i = 2$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	11.00	36%	7%	11.46	58%	3%
box	10.30	72%	8%	8.93	78%	7%
cycl	34.20	52%	12%	34.02	72%	3%
eleg	28.88	38%	3%	29.57	54%	10%
prmc	12.91	70%	12%	13.15	86%	8%
ball	2.75	76%	3%	2.85	92%	3%
volv	9.57	56%	3%	9.53	98%	3%
<b>Total</b>	15.66	57%	7%	15.64	77%	5%

Table 8.19: Results obtained by MSFIT with  $n_i = 2$ .

$n_i = 3$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	16.20	48%	8%	15.89	70%	8%
box	13.99	80%	8%	13.12	88%	8%
cycl	47.84	70%	6%	51.26	72%	8%
eleg	42.32	48%	5%	43.93	68%	8%
prmc	19.61	84%	7%	20.15	72%	5%
ball	3.85	76%	3%	4.26	94%	3%
volv	14.21	56%	6%	14.81	100%	0%
<b>Total</b>	22.58	66%	6%	23.34	81%	6%

Table 8.20: Results obtained by MSFIT with  $n_i = 3$ .

$n_i = 4$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	20.31	46%	7%	20.97	54%	17%
box	18.87	84%	7%	18.06	88%	5%
cycl	61.75	76%	7%	64.01	86%	3%
eleg	57.26	48%	5%	57.11	62%	8%
prmc	25.45	74%	6%	24.92	88%	7%
ball	5.15	76%	8%	5.61	92%	3%
volv	18.71	62%	3%	20.14	100%	0%
<b>Total</b>	29.64	67%	6%	30.12	81%	6 %

Table 8.21: Results obtained by MSFIT with  $n_i = 4$ .

$n_i = 5$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	24.60	42%	3%	25.76	62%	13%
box	22.81	82%	12%	21.76	90%	4%
cycl	76.76	62%	10%	79.68	86%	8%
eleg	68.67	50%	9%	71.21	60%	8%
prmc	30.76	90%	9%	31.46	98%	3%
ball	6.48	82%	10%	7.03	96%	3%
volv	22.39	70%	6%	25.62	98%	3%
<b>Total</b>	36.07	68%	9%	37.50	84%	6 %

Table 8.22: Results obtained by MSFIT with  $n_i = 5$ .

$n_i = 6$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	28.75	58%	12%	29.67	70%	12%
box	27.03	92%	5%	25.00	94%	3%
cycl	92.80	74%	3%	93.81	92%	5%
eleg	82.20	48%	10%	85.25	68%	9%
prmc	36.97	80%	6%	35.31	92%	5%
ball	7.47	86%	6%	8.21	98%	3%
volv	28.09	66%	8%	30.75	100%	0%
<b>Total</b>	43.33	72%	7%	44.00	88%	5 %

Table 8.23: Results obtained by MSFIT with  $n_i = 6$ .

$n_i = 7$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	32.37	58%	10%	33.77	78%	9%
box	30.84	92%	5%	27.93	94%	6%
cycl	104.20	72%	5%	109.73	96%	3%
eleg	92.60	52%	7%	94.47	62%	9%
prmc	42.39	80%	10%	43.56	92%	5%
ball	8.73	84%	6%	9.45	98%	3%
volv	32.05	64%	10%	35.49	100%	0%
<b>Total</b>	49.02	72%	8%	50.63	89%	5 %

Table 8.24: Results obtained by MSFIT with  $n_i = 7$ .

$n_i = 8$

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	35.52	68%	3%	38.02	76%	6%
box	32.90	84%	6%	30.63	92%	3%
cycl	117.64	82%	8%	121.63	80%	6%
eleg	107.28	40%	8%	108.08	58%	7%
prmc	46.98	78%	3%	46.62	90%	4%
ball	9.39	82%	5%	9.97	98%	3%
volv	35.85	58%	8%	39.49	100%	0%
<b>Total</b>	55.08	70%	6%	56.35	85%	4 %

Table 8.25: Results obtained by MSFIT with  $n_i = 8$ .

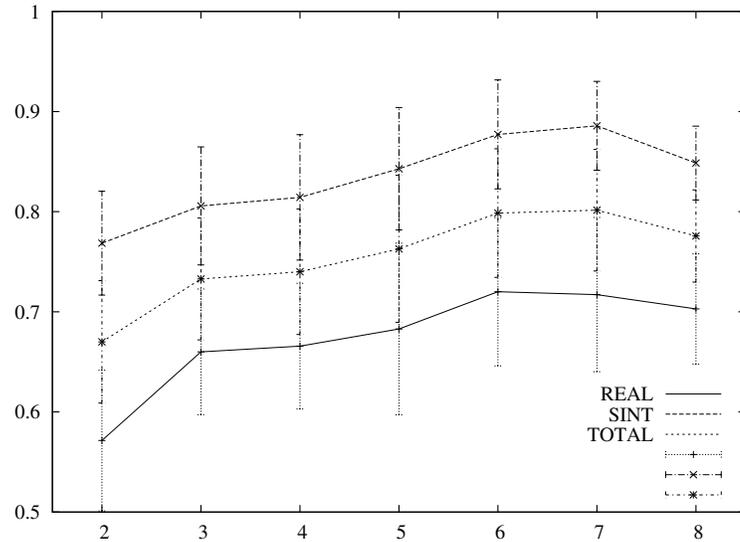


Figure 8.9: The effect of the parameter that determine the number of iterations  $n_i$  on the hit rate.

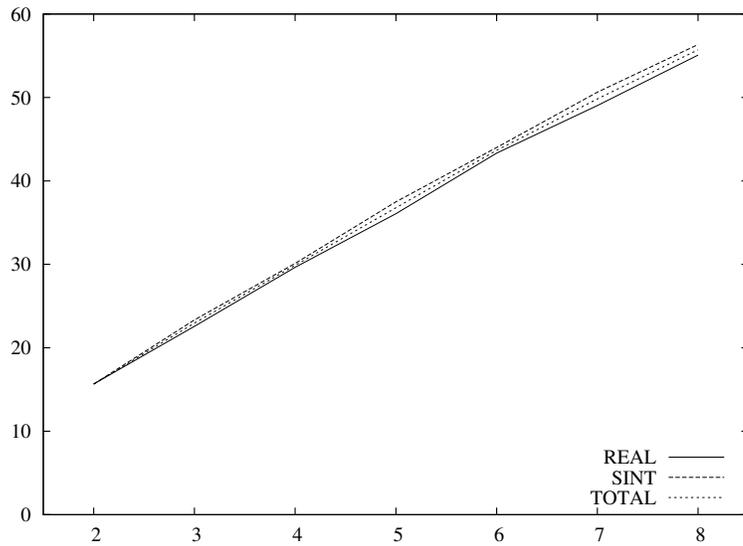


Figure 8.10: The effect of the parameter that determine the number of iterations  $n_i$  on the computational cost.

### 8.5.5 Effect of the queue size

In this subsection we test the effect of the internal parameter  $\alpha$  that defines the queue size at each scale (by the formula  $L_k = t + \lceil \alpha k \rceil$ ). This function is illustrated in the table below 8.26. For this test we vary  $\alpha$  between 0.0 and 2.0, for a fixed input parameter  $t = 1$ . See tables 8.27, 8.28, 8.29, 8.30, 8.31 8.32, 8.33 and figures 8.11 and 8.12. As the figures show, the hit rate initially grows with  $\alpha$  but then becomes almost independent of  $\alpha$  once  $\alpha \geq 0.5$  while the cost appears to grow linearly with  $\alpha$ . Therefore we chose  $\alpha = 0.5$ .

		scale									
		0	1	2	3	4	5	6	7	8	
$\alpha = 0.0$	$L_k$	1	1	1	1	1	1	1	1	1	
$\alpha = 0.3$		1	2	2	2	3	3	3	4	4	
$\alpha = 0.5$		1	2	2	3	3	4	4	5	5	
$\alpha = 1.0$		1	2	3	4	5	6	7	8	9	
$\alpha = 1.5$		1	3	4	6	7	9	10	12	13	
$\alpha = 2.0$		1	3	5	7	9	11	13	15	17	
$\alpha = 2.5$		1	4	6	9	11	14	16	19	21	

Table 8.26: The influence of  $\alpha$  on the size of  $L_k$ .

$\alpha = 0.0$						
dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	8.38	22%	5%	8.95	32%	5%
box	9.14	92%	3%	8.48	84%	10%
cycl	9.98	40%	8%	8.62	36%	11%
prmc	10.13	42%	7%	10.03	54%	13%
ball	2.31	66%	3%	2.79	98%	3%
volv	7.69	60%	8%	8.92	86%	6%
<b>Total</b>	7.94	54%	6%	7.97	65%	8 %

Table 8.27: Results obtained by MSFIT with  $\alpha = 0.0$ .

$\alpha = 0.5$ 

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	19.60	48%	10%	20.92	64%	13%
box	18.72	90%	9%	17.47	98%	3%
cycl	21.08	70%	8%	21.89	76%	10%
prmc	24.68	78%	5%	24.25	92%	5%
ball	5.20	74%	6%	5.93	98%	3%
volv	18.20	64%	7%	20.09	100%	0%
<b>Total</b>	17.91	71%	8%	18.42	88%	6 %

Table 8.28: Results obtained by MSFIT with  $\alpha = 0.5$ . $\alpha = 0.75$ 

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	24.86	58%	11%	25.21	72%	8%
box	22.55	88%	7%	20.84	96%	3%
cycl	27.07	70%	4%	27.04	82%	3%
prmc	29.01	82%	8%	29.38	92%	8%
ball	6.37	84%	3%	6.74	94%	3%
volv	21.83	64%	6%	24.37	98%	3%
<b>Total</b>	21.95	74%	7%	22.26	89%	5 %

Table 8.29: Results obtained by MSFIT with  $\alpha = 0.75$ . $\alpha = 1.0$ 

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	25.23	60%	9%	25.88	68%	5%
box	22.25	90%	6%	21.45	88%	5%
cycl	27.00	70%	15%	27.59	80%	10%
prmc	29.99	82%	7%	30.71	90%	6%
ball	6.39	86%	3%	6.67	100%	0%
volv	23.41	62%	5%	25.21	98%	3%
<b>Total</b>	22.38	75%	8%	22.92	87%	5 %

Table 8.30: Results obtained by MSFIT with  $\alpha = 1.0$ .

$\alpha = 1.25$ 

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	32.29	64%	11%	33.00	70%	8%
box	29.55	76%	7%	28.29	92%	3%
cycl	38.22	66%	7%	38.03	86%	3%
prmc	41.94	82%	7%	40.71	94%	3%
ball	8.35	78%	9%	8.92	100%	0%
volv	30.35	68%	5%	32.92	100%	0%
<b>Total</b>	30.12	72%	8%	30.31	90%	3 %

Table 8.31: Results obtained by MSFIT with  $\alpha = 1.25$ . $\alpha = 1.5$ 

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	34.92	60%	9%	36.53	78%	5%
box	32.88	78%	8%	29.94	96%	3%
cycl	41.05	74%	3%	40.29	92%	7%
prmc	44.46	92%	3%	43.19	92%	3%
ball	8.30	82%	3%	9.14	98%	3%
volv	33.84	64%	7%	36.13	100%	0%
<b>Total</b>	32.57	75%	6%	32.54	93%	4 %

Table 8.32: Results obtained by MSFIT with  $\alpha = 1.5$ . $\alpha = 2.0$ 

dataset	REAL			SINT		
model	$N_R/10^6$	$H_A$	$H_D$	$N_R/10^6$	$H_A$	$H_D$
cyl	39.02	58%	8%	40.61	82%	3%
box	34.62	76%	3%	33.14	100%	0%
cycl	47.45	66%	10%	46.27	96%	3%
prmc	51.46	86%	3%	50.58	92%	3%
ball	9.26	82%	12%	9.77	90%	4%
volv	38.68	66%	7%	41.09	98%	3%
<b>Total</b>	36.75	72%	7%	36.91	93%	3 %

Table 8.33: Results obtained by MSFIT with  $\alpha = 2.0$ .

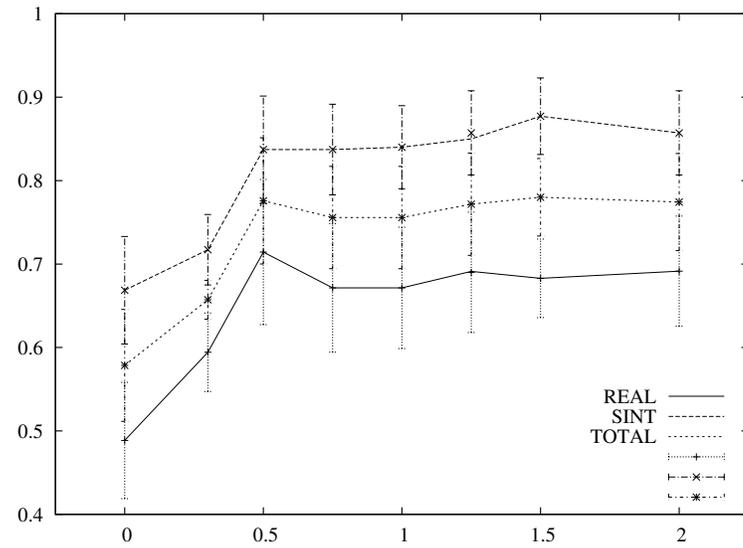


Figure 8.11: The effect of the parameter that determine the queue size  $\alpha$  on the hit rate.

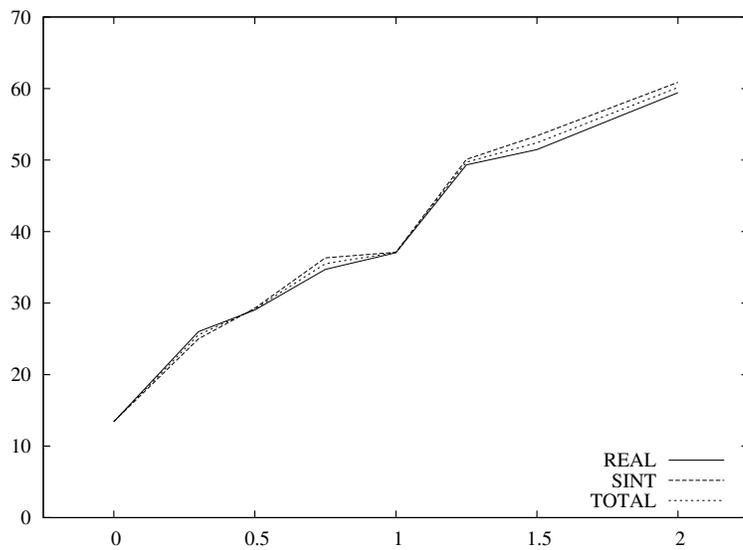


Figure 8.12: The effect of the parameter that determine the queue size  $\alpha$  on the computational cost.



# Chapter 9

## Conclusion

As the tests in Chapter 8 showed, our MSFIT algorithm can determine successfully the pose and deformations of complex tridimensional models (with a dozen free parameters and under significant deformation) on 70 – 75% of cases from a given 2D image. The tests shows its suitability to the intended applications, namely deformable biological models and optical microscope images.

### 9.1 Contributions of this thesis

The main contribution of this thesis is a *multiscale 3D-2D fitting algorithm* (MSFIT) to perform this task. The algorithm can be adapted for other tasks that require matching a 3D model of deformable organism to a 2D image, such as finding all instances of a specified organism in a given image; identifying an imaged organism among a collection of known organisms; and locating all images from a given collection where a specified organism appears.

The MSFIT algorithm also differs from most published 3D-2D model fitting algorithms in that it allows extreme deformations (limited only by the model itself). While several of those solutions use multiscale technique for efficiency [17, 19], MSFIT also uses it to cope with ambiguities (such as bilateral symmetry), occlusion and others complicating factors. To that end, MSFIT processes not one but several candidate fittings, in parallel through all scales, in a way similar to Zampieri’s multiscale image retrieval method [58, 57].

Another feature of our method is that it does not require the identification of salient points or lines in the image, and is insensitive to lighting and color variations. This is a significant difference over most other published methods [48, 19, 17]. Instead, MSFIT uses the outline and other silhouette edges of the projected object, with a special normalized gradient metric.

Our implementation of the MSFIT algorithm also uses a new nonlinear optimiza-

tion method, whose core is a quadratic minimizer using a divided-edge simplex sampling schema. Key features of this method are the automatic detection of irrelevant parameters and the automatic choice of sampling step for each relevant parameter.

## 9.2 Future work

The running time of the algorithm is still fairly high (2 minutes per image with the most complex models). We believe this can be speeded up considerably by further tuning of the internal parameters and of the details of the optimization step. Other future work consists in apply multi-objective optimization to obtain the best internal parameters.

# Appendix A

## Nonlinear optimization method

In this chapter we present a method for nonlinear optimization that was created by us that is based on quadratic interpolation, on the search for stationary point and edge-divided simplex.

### A.1 Function optimization by the edge-divided simplex method

#### A.1.1 Quadratic interpolation

We consider first the problem of determining a quadratic function  $Q : \mathbb{R}^n \rightarrow \mathbb{R}$  from given values at certain sample points. The function can be written in matrix form as

$$Q(x) = x^\top A x + x^\top B + C \quad (\text{A.1})$$

where  $A$  is a symmetric  $n \times n$  matrix,  $B$  is a column  $n$ -vector and  $C$  is a constant. The coefficients  $A$ ,  $B$  and  $C$  have  $m = (n + 1)(n + 2)/2$  unknown elements, and therefore they could be obtained in principle from the values of  $Q(q_i)$  of  $Q$  at  $m$  sampling points  $q_i$ ,  $i \in \{0..m - 1\}$  by solving a system with  $m$  linear equations. However, depending on the placement of the sampling points  $q_i$ , the system may be indeterminate or numerically unstable. We show that by proper selection of the sample points, the function can be determined directly by a simple stable formula.

Our method samples  $Q$  at the corners and edge midpoints of an arbitrary simplex in  $\mathbb{R}^n$ . We start with a simplex  $S = (p_0, p_1, \dots, p_n)$  with  $n + 1$  points not all in the same hyper-plane. Then we compute the midpoints  $p_{ij} = (p_i + p_j)/2$  between the vertices  $p_i$  and  $p_j$ , for  $0 \leq i, j \leq n$ . Note that  $p_{ii} = p_i$  and  $p_{ij} = p_{ji}$ , so there are only  $n(n - 1)/2$  midpoints to be computed. In total we have the minimum required number  $m = (n + 2)(n + 1)/2$

of sample points. See figure A.1.

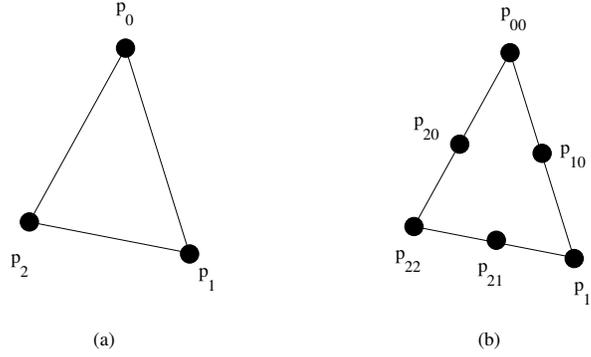


Figure A.1: A simplex of  $\mathbb{R}^2$  and the derived sampling points  $p_{ij}$ .

Next we express the function  $Q$  in terms of barycentric coordinates relative to the simplex  $S$ . Namely, we write

$$Q(x) = \tilde{Q}(z) = z^\top T z \quad (\text{A.2})$$

where  $z$  is the column vector of  $n + 1$  barycentric coordinates of the point  $x$  relative to the simplex vertices  $p_0, p_1, \dots, p_n$ . That is,  $z = (z_0, z_1, \dots, z_n)$  such that

$$\sum_{i=0}^n z_i = 1 \quad \text{and} \quad \sum_{i=0}^n z_i p_i = x \quad (\text{A.3})$$

The vector  $z$  is an element of the *canonical affine  $n$ -space*  $\mathbb{A}^n$ , the set of all vectors of  $\mathbb{R}^{n+1}$  that have unit sum.

Let  $y_{ij}$  be the given function value at point  $p_{ij}$ . Then the interpolating homogeneous quadratic function  $\tilde{Q}$  can be obtained by the formula

$$\tilde{Q}(z) = \sum_{i=0}^n y_{ii} \phi_i(z) + \sum_{\substack{i,j=0 \\ i \neq j}}^n y_{ij} \psi_{ij}(z) \quad (\text{A.4})$$

where  $\phi_i$  and  $\psi_{ij}$  are the elements of the quadratic interpolating basis for the sampling points  $p_{ii}$  and  $p_{ij}$  in barycentric coordinates. Namely,

$$\phi_i(z) = 2 z_i (z_i - 1/2) \quad (\text{A.5})$$

$$\psi_{ij}(z) = 2 z_i z_j \quad (\text{A.6})$$

Observe that

$$\phi_i(p_{rs}) = (i = r)(i = s) \quad (\text{A.7})$$

for all  $i, r, s \in \{0 \dots m\}$ , and

$$\psi_{ij}(p_{rs}) = ((i = r)(j = s) + (i = s)(j = r))/2 \quad (\text{A.8})$$

for all  $i, j, r, s \in \{0 \dots m\}$  with  $i \neq j$ . Therefore the matrix  $T$  in formula (A.2) is given by

$$T_{ij} = \begin{cases} y_i & \text{if } i = j \\ 2y_{ij} - (y_i + y_j)/2 & \text{if } i \neq j \end{cases} \quad (\text{A.9})$$

### A.1.2 Quadratic optimization

Now consider the problem of finding the stationary point (minimum, maximum, or saddle point) of a quadratic function  $Q : \mathbb{R}^n \rightarrow \mathbb{R}$ . If  $Q$  is given by formula (A.1), the solution is the point  $x^* \in \mathbb{R}^n$  such that  $\nabla Q(x^*) = 0$  where

$$\nabla Q(x^*) = 2A x^* + B \quad (\text{A.10})$$

If  $Q$  is expressed in barycentric coordinates by formula (A.2), the minimum, maximum or saddle point of the corresponding function  $\tilde{Q}$  is the point  $z^* \in \mathbb{A}^n$  such that  $\nabla \tilde{Q}(z^*) = (\lambda, \lambda, \dots, \lambda)$  for some  $\lambda \in \mathbb{R}$ , where  $\nabla \tilde{Q}$  is the gradient of  $\tilde{Q}$  in  $\mathbb{R}^{n+1}$ ; that is

$$\nabla \tilde{Q}(z^*) = 2T z^* \quad (\text{A.11})$$

Therefore the point  $z^*$  can be found by solving the linear system  $Mz^* = D$  where  $M$  is the matrix with  $n + 2$  lines and  $n + 2$  columns

$$M = \begin{pmatrix} & & & & & & & 1 \\ & & & & & & & 1 \\ & & & & & & & 1 \\ & & & & & & & 1 \\ & & & & & & & 1 \\ & & & & & & & 1 \\ & & & & & & & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$D$  is the column  $(n + 2)$ -vector  $(0, 0, \dots, 0, 1)^\top$ , and  $z^*$  is the column vector of the  $(n + 2)$  unknowns  $(z_0^*, z_1^*, \dots, z_n^*, \lambda)^\top$ . Then the point  $x^*$  can be computed from  $z^*$  by the formula (A.3).

This method will probably fail if the matrix  $A$  of formula (A.1) has a null or a very small eigenvalue; that is, if there is a principal direction along which the original function  $Q$  is approximately affine (first degree polynomial). In those cases the matrix  $M$  will be singular or near-singular.

### A.1.3 Iterative quadratic optimization

Suppose now that we want to find a local stationary point (minimum, maximum, or saddle point)  $x^*$  of a nonlinear function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  (*the goal function*) that is twice differentiable. Suppose also we have an initial guess  $x^{(0)}$  for the solution and an upper bound  $\delta^{(0)}$  for the distance from  $x^{(0)}$  to the true solution  $x^*$ . We can do so by iterating the quadratic optimization algorithm of section A.1.2. See algorithm 2. The source code available at [9].

---

**Procedure 2** *DivSmpMin*( $f, x, \delta, \epsilon, t, \theta, \alpha, \beta, \sigma$ )

---

**Requires:** A function  $f$ ; the number  $n$  of variables, an initial guess  $x \in \mathbb{R}^n$ ; the initial error estimate  $\delta$ ; the error tolerance  $\epsilon$ ; the maximum number of iterations  $t$ ; the relative simplex radius  $\theta$ ; the uncertainty adjustment factors  $\alpha$  and  $\beta$ ; and the optimization direction  $\sigma \in \{-1, 0 + 1\}$ .

**Returns:** The local optimum point  $x^*$  of  $f$  and the new uncertainty estimate  $\delta$

```

1: while  $t \geq 0$  do
2:    $r \leftarrow \theta \delta$ ;
3:    $(p, y) \leftarrow \text{SimplexSample}(f, n, x, r)$ ;
4:    $z^* \leftarrow \text{OptQuad}(f, n, y)$ ;
5:    $x^* \leftarrow \sum_{i=0}^{n+1} z_i^* p_i$ ;
6:    $d \leftarrow \|x - x^*\|$ ;
7:   if  $(d > \delta)$  then  $x^* \leftarrow x + \delta(x - x^*)/d$ ;
8:   if  $(\sigma = +1)$  then
9:     Let  $y_{ij}$  the maximum sample function value;
10:    if  $(f(x^*) < y_{ij})$  then  $x^* \leftarrow p_{ij}$ ;  $d \leftarrow \|x - x^*\|$ ;
11:  if  $(\sigma = -1)$  then
12:    Let  $y_{ij}$  the minimum sample function value;
13:    if  $(f(x^*) > y_{ij})$  then  $x^* \leftarrow p_{ij}$ ;  $d \leftarrow \|x - x^*\|$ ;
14:   $\delta^* = \min\{\alpha\delta, \beta\sqrt{d^2 + n}\}$ 
15:   $x \leftarrow x^*$ ;  $\delta \leftarrow \delta^*$ ;  $t \leftarrow t - 1$ ;
16:  if  $(\delta^* < \epsilon)$  then return  $(x, \delta)$ ;
17: end while

```

---

At each step, our algorithm first generates a simplex  $S$  with circumradius  $\theta\delta$  surrounding the current guess  $x$ , where  $\theta$  is some fixed parameter usually smaller than 1. Then we interpolate the goal function  $f$  by a quadratic function  $Q$  at the vertices and midpoints of  $S$ , as described in section A.1.1. Next we compute the stationary point  $x^*$  of  $Q$  as described in section A.1.2. If the new guess is adequate, we compute a new uncertainty  $\delta$  based on the current  $\delta$  and the distance  $\|x^* - x\|$ , and we set the next guess  $x$  to  $x^*$ . This process is repeated until the estimated uncertainty  $\delta$  is less than a prescribed tolerance  $\epsilon$ , or a fixed computation budget is exhausted.

The input parameter  $\sigma$  indicates whether the algorithm should search for a maximum ( $\sigma = +1$ ), a minimum ( $\sigma = -1$ ) or any stationary point ( $\sigma = 0$ ). The new guess  $x^*$  is considered to be inadequate if the length  $\|x^* - x\|$  of the displacement is greater than  $\delta$ . In that case we truncate to displacement to length  $\delta$  preserving its direction (step 7). When looking for a minimum ( $\sigma = -1$ ) or a maximum ( $\sigma = +1$ ), the new guess  $x^*$  is inadequate also if its function value is not better than the sample values. If so, we set  $x^*$  to the point  $p_{ij}$  that has the best value of  $f(p_{ij})$  and adjust  $x$  according (steps 8 to 13).

To justify this method we observe that if the vertices of  $S$  are sufficiently close to the optimum  $x^*$ , then  $Q$  is too close to  $f$ ; and if  $f$  is positive definite at  $x^*$ , then the stationary point  $x^*$  of  $Q$  will be close to true optimum  $x^*$  of  $f$ . In fact, convergence is expected to be quadratic as in Newton's root-finding method, for the same reasons.

This method generally fails if the quadratic interpolant  $Q$  is not a good match to the goal function; either because function is "noisy" or because the initial guess is not close enough to the minimum, or because the initial simplex radius  $\theta\delta$  is too large.

### A.1.4 Examples

To illustrate the method we test it with various functions and various starting guesses. For each test we ran the algorithm for several starting guesses located on a regular array around the true minimum. We show the results as an "arrow plot" (such as figure A.3) where each arrow, representing one run of algorithm, connects the starting guess to the final result of that run. We also show a trace of a single run of the algorithm (such as figure A.4) where each arrow connects one intermediate guess  $x$  to the next guess  $x^*$ , and each circle has center at the current guess  $x$  and radius equal to the uncertainty  $\delta$ . In all tests, we set  $\alpha = 0.9$ ,  $\beta = 0.9$ ,  $\theta = 0.5$ ,  $\epsilon = 0.001$  and  $t = 10$ .

#### Quadratic function

For this test, the goal function is a quadratic polynomial in two variables [27].

$$f_1(x_0, x_1) = (x_0 + 2x_1 - 7)^2 + (2x_0 + x_1 - 5)^2 \quad (\text{A.12})$$

The minimum  $x^*$  of  $f_1$  is the point  $(1, 3)$  where  $f_1$  has the value 0. As expected for a quadratic function the algorithm finds the minimum in a single iteration.

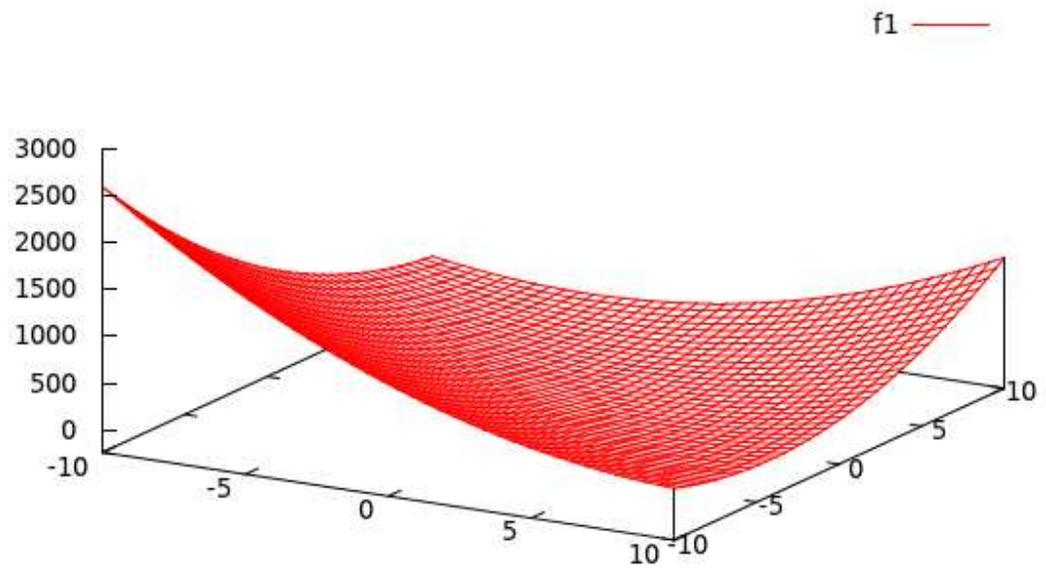


Figure A.2: Graph of the function  $f_1$ . The plotted region is  $[-11, 11] \times [-11, 11]$ .

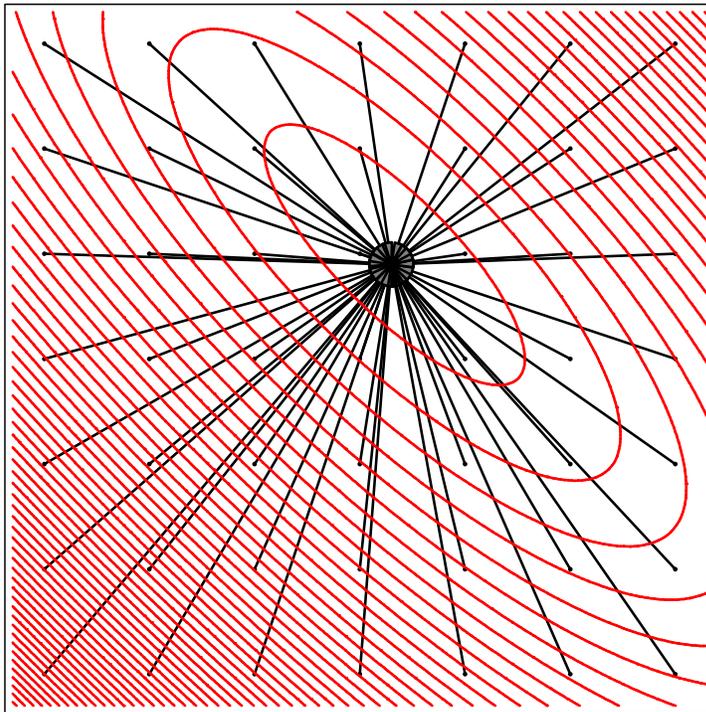


Figure A.3: The initial guesses and final results of our algorithm applied to  $f_1$  at several points in the square  $[-10, 10] \times [-10, 10]$ . The plotted region is  $[-11, 11] \times [-11, 11]$ . The uncertainty initial  $\delta$  was set to 15.

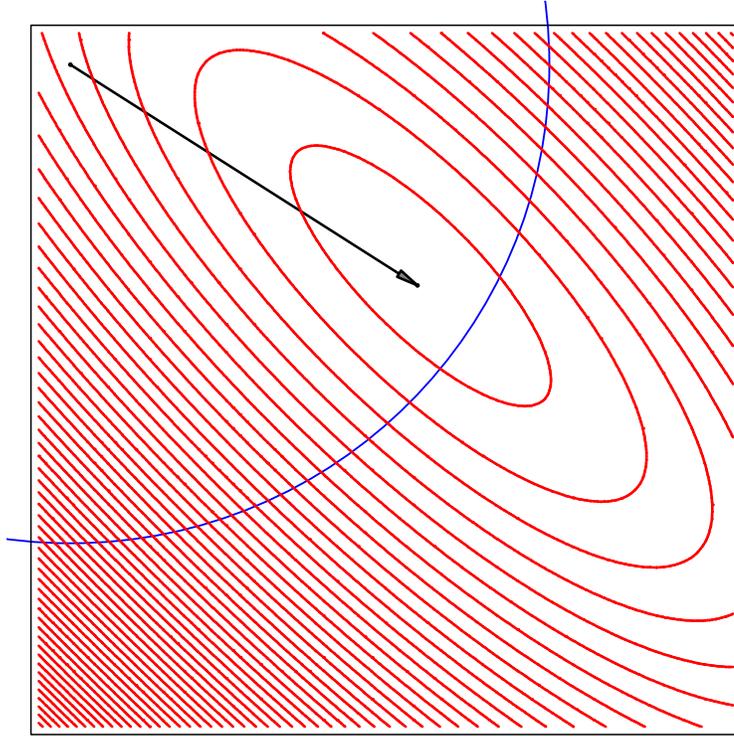


Figure A.4: Behavior of our algorithm applied to  $f_1$  with initial guess  $x = (-10, 10)$  and uncertainty  $\delta = 15$ . The plotted region is  $[-11, 11] \times [-11, 11]$ .

### Trigonometric polynomial

For this test, the goal function  $f_2$  [27] is a sum of sinusoidal waves along  $x_0$  and  $x_1$ , with various frequencies and phases

$$\begin{aligned}
 f_2(x_0, x_1) = & \\
 & [\cos(2x_1 + 1) + 2 \cos(3x_1 + 2) + \\
 & 3 \cos(4x_1 + 3) + 4 \cos(5x_1 + 4) + 5 \cos(6x_1 + 5)] \\
 & [\cos(1) + 2 \cos(x_0 + 2) + \\
 & 3 \cos(2x_0 + 3) + 4 \cos(3x_0 + 4) + 5 \cos(4x_0 + 5)]
 \end{aligned} \tag{A.13}$$

The minimum point  $x^*$  of  $f_2$  is approximately  $(4.97648, 4.85806)$ , where has value  $\approx -176.542$ .

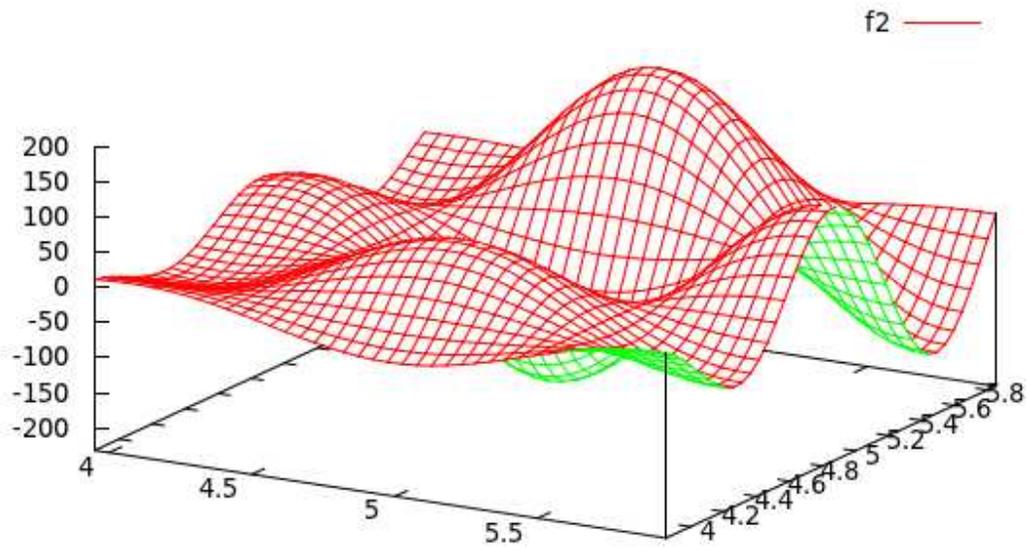


Figure A.5: Graph of the function  $f_2$ . The plotted regions is  $[3.95, 5.95] \times [3.85, 5.85]$ .

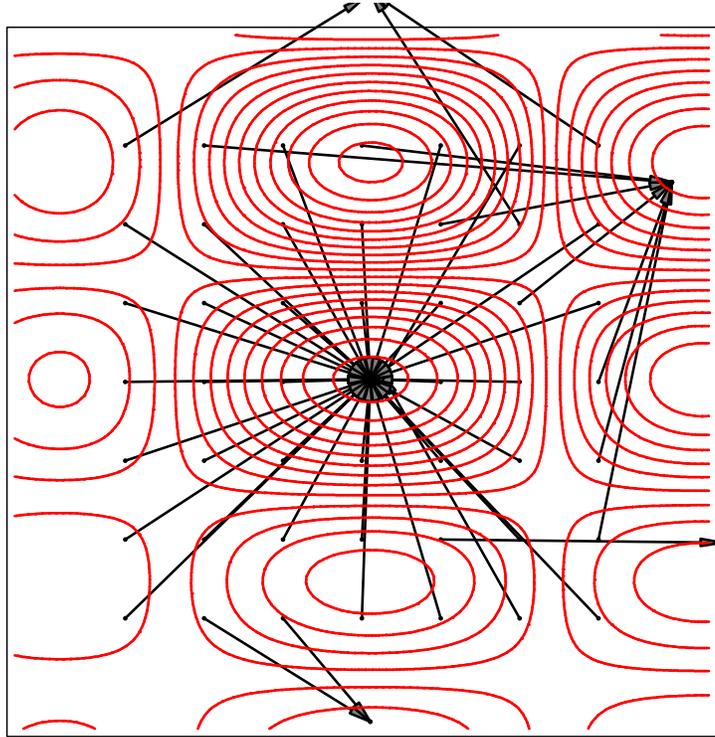


Figure A.6: The initial guesses and final results of our algorithm applied to  $f_2$  at several points in the square  $[4.2, 5.7] \times [4.1, 5.6]$ . The initial uncertainty  $\delta$  was set to 0.85. The plotted region is  $[3.95, 5.95] \times [3.85, 5.85]$ .

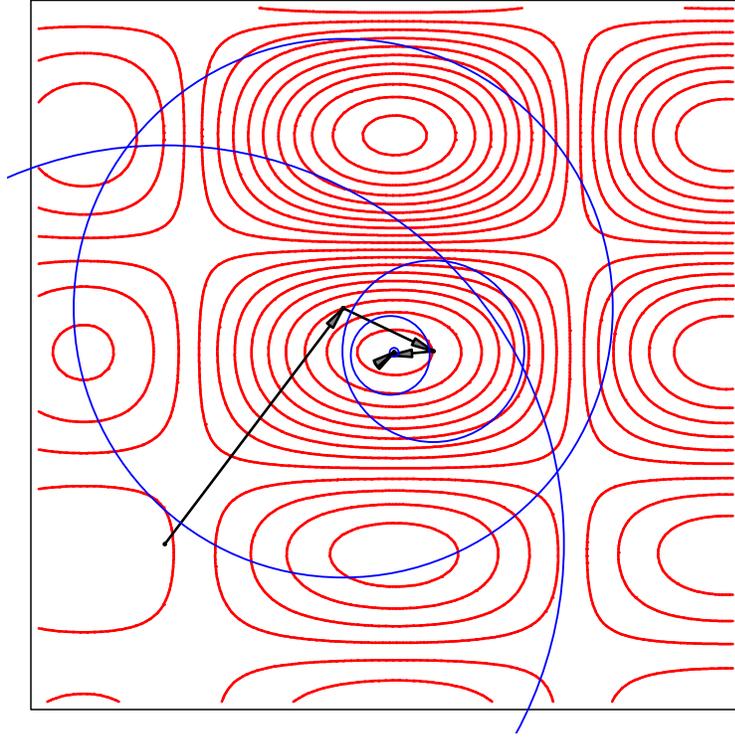


Figure A.7: Behavior of our algorithm applied  $f_2$  at initial guess  $x = (4.25, 4.25)$  and uncertainty  $\delta = 1.25$ . The plotted region is  $[3.95, 5.95] \times [3.85, 5.85]$ .

### Sextic polynomial

For this test, the goal function  $f_3$  is the “six-hump camel back function,” [10] a polynomial of the sixth degree

$$f_3(x_0, x_1) = \left(4 - \frac{21}{10}x_0^2 + \frac{x_0^4}{3}\right)x_0^2 + x_0x_1 + (-4 + 4x_1^2)x_1^2 \quad (\text{A.14})$$

This function has two local minima, approximately at  $(-0.0898, 0.7126)$  and  $(0.0898, -0.7126)$ , both with value  $\approx -1.0316$ .

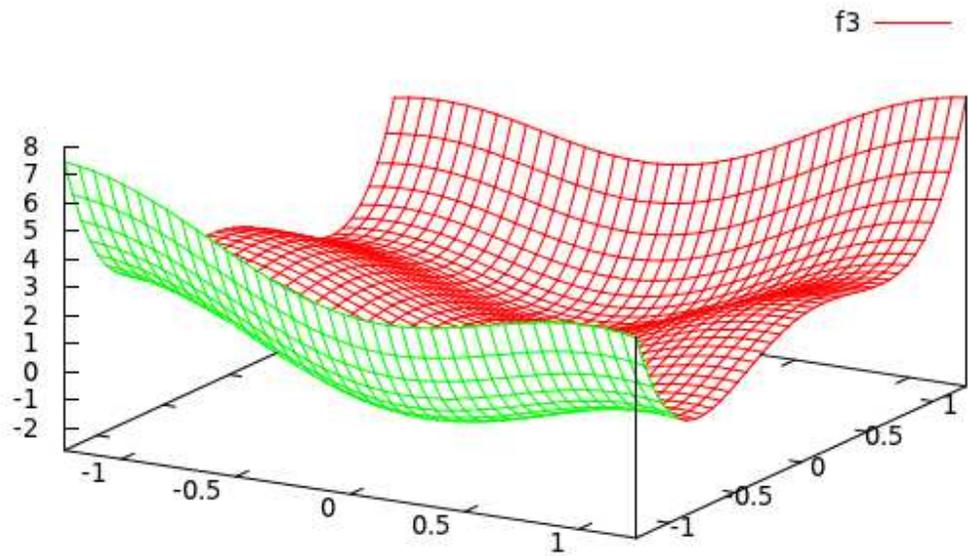


Figure A.8: Graph of the function  $f_3$ . The plotted region is  $[-1.25, 1.25] \times [-1.25, 1.25]$ .

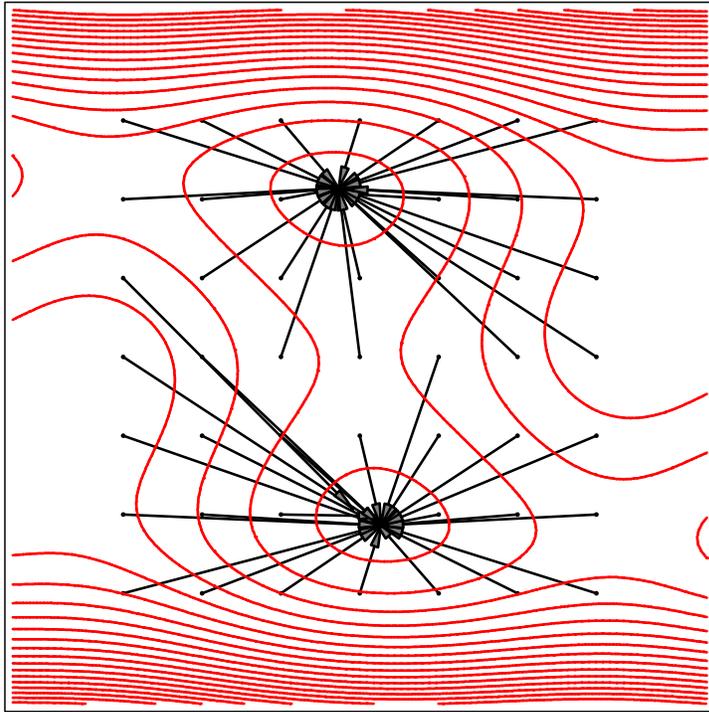


Figure A.9: The initial guesses and final results of our algorithm applied to  $f_3$  at several points in the square  $[-1, 1] \times [-1, 1]$ , with initial uncertainty  $\delta$  was set to 0.85. The plotted region is  $[-1.25, 1.25] \times [-1.25, 1.25]$ .

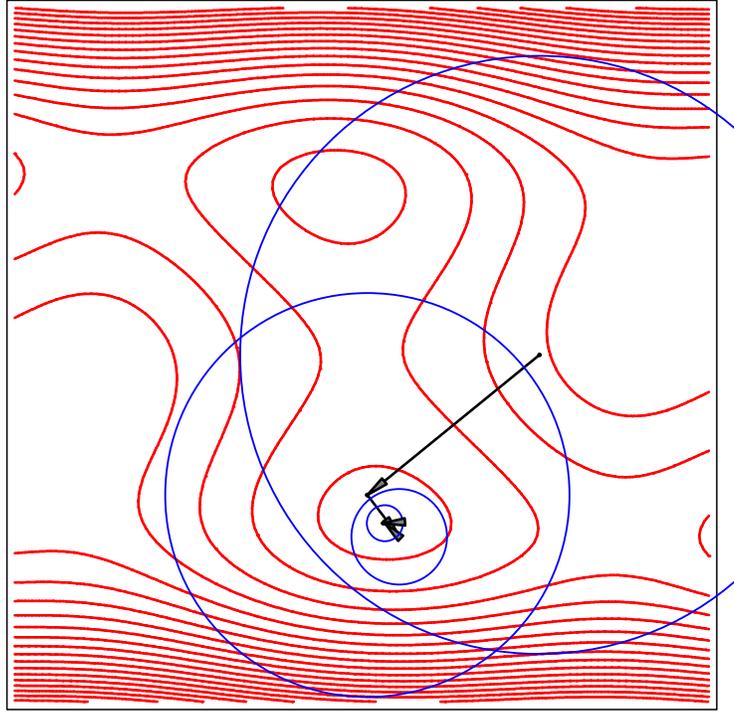


Figure A.10: Behavior of our algorithm applied to  $f_3$  with initial guess  $x = (0.75, 0)$ , uncertainty  $\delta = 1.1$ . The plotted region is  $[-1.25, 1.25] \times [-1.25, 1.25]$ .

### Trigonometric noise function

For this test, the goal function  $f_4$  is a sum of four sinusoidal waves of various directions and phases, and widely different frequencies

$$\begin{aligned}
 f_4(x_0, x_1) &= 2 \cos(3x_0 + 4x_1) & + & \cos(5x_0 - 2x_1) \\
 &+ \frac{1}{12} \cos(30x_0 + 12x_1) & + & \frac{1}{15} \cos(13x_0 - 27x_1)
 \end{aligned} \tag{A.15}$$

This function has several local minima; some of them are approximately located at  $(0.6829, 0.22)$ ,  $(-0.6829, -0.22)$ ,  $(1.218, 1.418)$  and  $(-1.218, -1.418)$ , with slightly different function values close to  $-3.041$ .

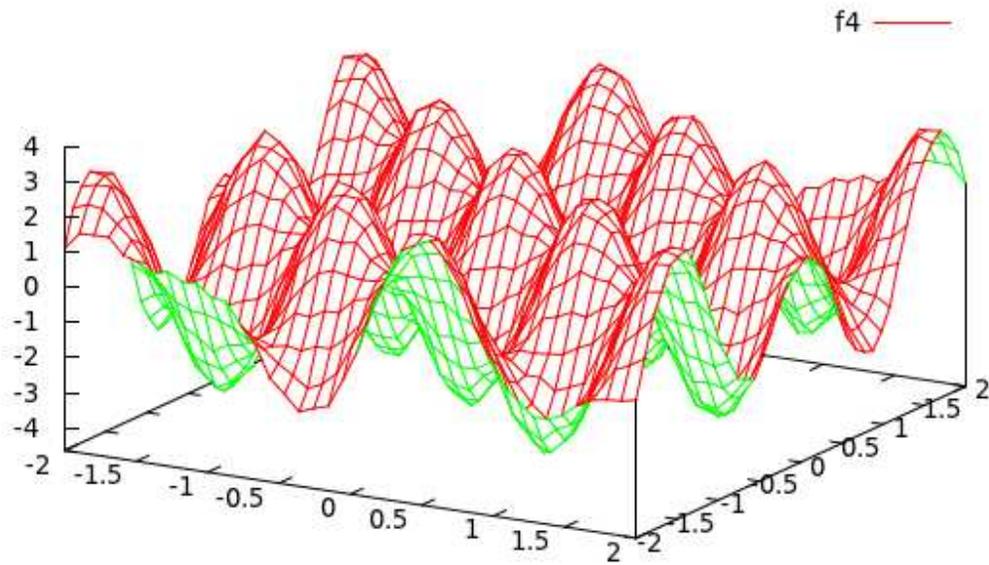


Figure A.11: Graph of the function  $f_4$ . The plotted region is  $[-2, 2] \times [-2, 2]$ .

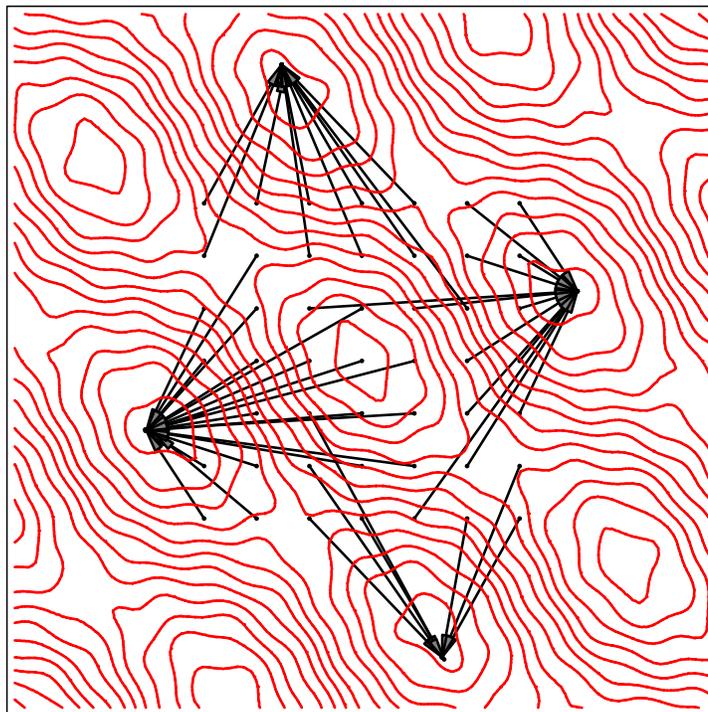


Figure A.12: The initial guesses and final results of our algorithm applied to  $f_4$  at several points in the square  $[-1, 1] \times [-1, 1]$ . The initial uncertainty  $\delta$  was set to 0.95. The plotted region is  $[-2, 2] \times [-2, 2]$ .

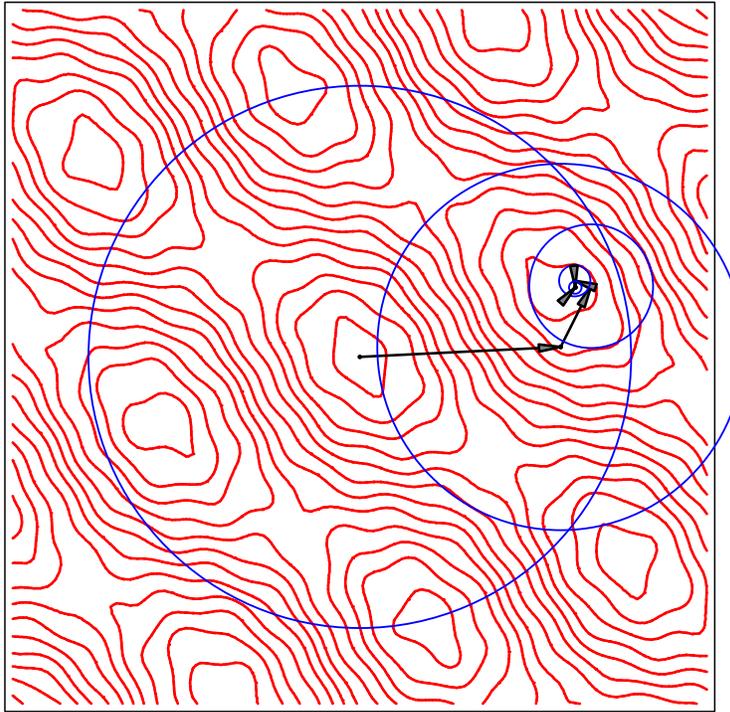


Figure A.13: Behavior of our algorithm applied to  $f_4$  with initial guess  $x = (0, 0)$ , uncertainty  $\delta = 0.85$ . The plotted region is  $[-2, 2] \times [-2, 2]$ .

# Bibliography

- [1] Image copied from website <http://www.wikipedia.org>.
- [2] Image copied from website <http://www.marksimmons.org/microscope/microimages/pages/paramecium2.htm>.
- [3] Image copied from website <http://www.ncbi.nlm.nih.gov/projects/genome/guide/nematode/index.html>.
- [4] Image copied from website <http://www.kiwicrossing.com/drrussell/projects.html>.
- [5] Image copied from website <http://www.appslabs.com.au/daphnia.htm>.
- [6] Image copied from website [http://botit.botany.wisc.edu/images/130/symbiosis\\_images/Paramecium\\_w\\_Chlorella\\_MC.html](http://botit.botany.wisc.edu/images/130/symbiosis_images/Paramecium_w_Chlorella_MC.html).
- [7] Image copied from website [http://www.ufmt.br/bionet/conteudos/01.09.04/tip\\_mic.htm](http://www.ufmt.br/bionet/conteudos/01.09.04/tip_mic.htm).
- [8] Image rendered by the software POV-Ray. Available at <http://www.povray.org/>.
- [9] The source code available at <http://www.liv.ic.unicamp.br/~danillorp/0n1/>.
- [10] Molga, Marcin and Smutnicki, Czesaw. Test functions for optimization needs (2005). <http://www.bioinformaticslaboratory.nl/twikidata/pub/Education/NBICResearchSchool/Optimization/VanKampen/BackgroundInformation/TestFunctions-Optimization.pdf>.
- [11] Bouchra Abboud, Franck Davoine, and Mo Dang. Facial expression recognition and synthesis based on an appearance model, 2004.
- [12] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

- [13] Bruno Arnaldi, Thierry Priol, and Kadi Bouatouch. A new space subdivision method for ray tracing csg modelled scenes. *The Visual Computer*, 3(2):98–108, 1987.
- [14] Duret L et al. Aury JM, Jaillon O. Global trends of whole-genome duplications revealed by the ciliate paramecium tetraurelia. In *Nature* 444, pages 171–178, November 2006.
- [15] Serge Belongie, Jitendra Malik, and Jan Puzicha. Matching shapes, 2001.
- [16] Stefano Berretti, Alberto Del Bimbo, Pietro Pala, and Francisco Silva-Mata. Geodesic distances for 3d-3d and 2d-3d face recognition. In *ICME'07*, pages 1515–1518, 2007.
- [17] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques, SIGGRAPH '99*, pages 187–194, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [18] Peter Blicher and Sebastien Roy. Fast lighting/rendering solution for matching a 2d image to a database of 3d models: "lightsphere", 2001.
- [19] Chun-Wei Chen and Chieh-Chih Wang. 3d active appearance model for aligning faces in 2d images. In *IEEE/RSJ International Conference on Robots and Systems (IROS)*, Nice, France, September 2008.
- [20] Ting Chen, Xiaoxu Wang, Dimitris N. Metaxas, and Leon Axel. 3d cardiac motion tracking using robust point matching and meshless deformable models. In *ISBI*, pages 280–283, 2008.
- [21] Chi-Wei Chu and Ramakant Nevatia. Real-time 3d body pose tracking from multiple 2d images. In *Proceedings of the 5th international conference on Articulated Motion and Deformable Objects, AMDO '08*, pages 42–52, Berlin, Heidelberg, 2008. Springer-Verlag.
- [22] Haili Chui and Anand Rangarajan. A new algorithm for non-rigid point matching. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2044, 2000.
- [23] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 484–498. Springer, 1998.

- [24] Helena Cristina da Gama Leitão and Jorge Stolfi. A multiscale method for the reassembly of two-dimensional fragmented objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(9):1239–1251, 2002.
- [25] Elisa de Cassia Silva. *Modelagem de deformacao do espaco 2.5D para estruturas biologicas*. Master thesis, UNICAMP, BRAZIL., 2011.
- [26] Elisa de Cassia Silva Rodrigues, Anamaria Gomide, and Jorge Stolfi. A User-editable  $C^1$ -Continuous 2.5D Space Deformation Method For 3D Models. Technical Report IC-11-14, Institute of Computing, University of Campinas, June 2011.
- [27] Luiz H. de Figueiredo, Ronald Van Iwaarden, and Jorge Stolfi. Fast interval branch-and-bound methods for unconstrained global optimization with affine arithmetic. Technical Report IC-97-08, Institute of Computing, Univ. of Campinas, June 1997.
- [28] Xinbo Gao, Ya Su, Xuelong Li, and Dacheng Tao. A review of active appearance models. *Trans. Sys. Man Cyber Part C*, 40(2):145–158, March 2010.
- [29] Lie Gu and Takeo Kanade. 3d alignment of face in a single image. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 1305–1312, Washington, DC, USA, 2006. IEEE Computer Society.
- [30] M.G. Danilouchkine H.J. Lamb J.H.C. Reiber B.P.F. Lelieveldt H.C. van Assen, R.J. van der Geest. Three-dimensional active shape model matching for left ventricle segmentation in cardiac ct. In *Proceedings of the SPIE 2003. Medical Imaging: Image Processing.*, United States, San Diego, 2003.
- [31] Changbo Hu, Jing Xiao, Iain Matthews, Simon Baker, Jeffrey Cohn, and Takeo Kanade. Fitting a single active appearance model simultaneously to multiple images. In *Proceedings of the British Machine Vision Conference*, September 2004.
- [32] Yumi Iwashita, Kurazu Ryo, Kozo Konishi, Masahiko Nakamoto, Hashizu Makoto, and Tsutomu Hasegawa. Fast alignment of 3d geometrical models and 2d grayscale images using 2d distance maps. *Syst. Comput. Japan*, 38:52–62, December 2007.
- [33] Mike Fornefett Karl, Karl Rohr, and H. Siegfried Stiehl. Elastic registration of medical images using radial basis functions with compact support. In *In Proc. of CVPR'99*, pages 402–407, 1999.
- [34] Timothy L. Kay and James T. Kajiya. Ray tracing complex scenes. In *SIGGRAPH*, pages 269–278, 1986.

- [35] Shian-Ru Ke, LiangJia Zhu, Jenq-Neng Hwang, Hung-I Pai, Kung-Ming Lan, and Chih-Pin Liao. Real-time 3d human pose estimation from monocular view with applications to event detection and video gaming. In *Proceedings of the 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS '10*, pages 489–496, Washington, DC, USA, 2010. IEEE Computer Society.
- [36] James Kennedy and Russell Eberhart. Particle swarm optimization, 1995.
- [37] Jeffrey M. Lane and Richard F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2(1):35–46, January 1980.
- [38] J. K. Lowry. . crustacea, the higher taxa: Description, identification, and information retrieval. In *Cyclopoida (Copepoda, Maxillipoda)*, October 1999.
- [39] Sean Luke. *Essentials of Metaheuristics*. Lulu, 2009. Available for free at <http://cs.gmu.edu/~sean/book/metaheuristics/>.
- [40] Tim Mcinerney, Tim Mcinerney, and Demetri Terzopoulos. Deformable models in medical image analysis: A survey, 1996.
- [41] Rodrigo Minetto. Reconhecimento de texto e rastreamento de objetos 2d/3d. *PhD Thesis.*, 2012.
- [42] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
- [43] Danillo Roberto Pereira. Representação e cálculo eficiente da iluminação global na síntese de imagem. Master’s thesis, Institute of Computing, University of Campinas, 2009.
- [44] Danillo Roberto Pereira, Jorge Stolfi, and Rafael Felipe Veiga Saracchini. A new method for nonlinear optimization. Technical Report IC-11-19, Institute of Computing, University of Campinas, November 2011.
- [45] J. H. Powers. Further studies in volvox, with descriptions of three new species. In *Transactions of the American Microscopical Society* 28, pages 141–175, 1908.
- [46] A. Sattar, Y. Aidarous, S. Le Gallou, and R. Segurier. Face alignment by 2.5d active appearance model optimized by simplex.
- [47] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986.

- [48] Yanchao Su, Haizhou Ai, and Shihong Lao. Multi-view face alignment using 3d shape model for view estimation. In Massimo Tistarelli and Mark S. Nixon, editors, *ICB*, volume 5558 of *Lecture Notes in Computer Science*, pages 179–188. Springer, 2009.
- [49] Celso T. N. Suzuki, Jancarlo F. Gomes, Alexandre X. Falcão, João P. Papa, and Sumie Hoshino-Shimizu. Automatic segmentation and classification of human intestinal parasites from microscopy images. *IEEE Trans. Biomed. Engineering*, 60(3):803–812, 2013.
- [50] Alexander Toshev, Ameesh Makadia, and Kostas Daniilidis. Shape-based object recognition in videos using 3d synthetic object models. In *CVPR*, pages 288–295. IEEE, 2009.
- [51] P. Viola and III W.M. Wells. Alignment by maximization of mutual information. *Computer Vision, IEEE International Conference on*, 0:16, 1995.
- [52] Turner Whitted. An improved illumination model for shaded display. In *Proceedings of the 6th annual conference on Computer graphics and interactive techniques, SIGGRAPH '79*, pages 14–, New York, NY, USA, 1979. ACM.
- [53] Turner Whitted. An improved illumination model for shaded display. *SIGGRAPH Comput. Graph.*, 13(2):14–, August 1979.
- [54] William Barry Wood. *The Nematode Caenorhabditis elegans*. Cold Spring Harbor Laboratory Press, 1988.
- [55] Nobuyuki Kita Yasuyo Kita. On accuracy of 3d localization obtained by aligning 3d model with observed 2d occluding edges. In *Asian Conference on Computer Vision*, 2002.
- [56] Chanjira Sinthanayothin Yootthana Mahitdharn. Registration between 3d object and video image for brackets alignment on dental cast. In *The 3rd International Symposium on Biomedical Engineering (ISBME 2008)*, 2008.
- [57] Carlos E. A Zampieri. *Recuperação de imagens multiescala intervalar*. Master thesis, UNICAMP, BRAZIL., 2010.
- [58] Carlos E. A. Zampieri and Jorge Stolfi. Image retrieval by multi-scale interval distance estimation. *17th International Conference on Systems, Signals and Image Processing*, 2010.
- [59] Carlos E. A. Zampieri and Jorge Stolfi. Image retrieval by multi-scale interval distance estimation. *Technical Report - Institute of Computing - UNICAMP*, 2010.

- [60] W. Zhao, D. Nister, and S. Hsu. Alignment of continuous video onto 3d point clouds. *Proc. of CVPR, Vol.2*, 27:964971, 2004.
- [61] Youding Zhu and Kikuo Fujimura. Bayesian 3d human body pose tracking from depth image sequences. In *ACCV (2)*, pages 267–278, 2009.